VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

VYHLEDÁVÁNÍ KONKRÉTNÍCH OSOB NA ZÁZNAMECH Z BEZPEČNOSTNÍCH KAMER

SEARCH FOR PEOPLE IN RECORDINGS FROM SECURITY CAMERAS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **MATOUŠ JEZERSKÝ**
AUTHOR

VEDOUCÍ PRÁCE **Doc. RNDr. PAVEL SMRŽ, Ph.D.**
SUPERVISOR

BRNO 2018

Zadání diplomové práce/18530/2017/xjezer01

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2017/2018

Zadání diplomové práce

Řešitel: **Jezerský Matouš, Bc.**
Obor: Inteligentní systémy
Téma: **Vyhledávání konkrétních osob na záznamech z bezpečnostních kamer**
Search for People in Recordings from Security Cameras
Kategorie: Umělá inteligence

Pokyny:

1. Seznamte se se způsoby vyhledávání tváří a s dostupnými implementacemi, např. http://docs.opencv.org/trunk/modules/contrib/doc/facerec/
2. Shromážděte datovou sadu pro průběžné testování systému.
3. Na základě získaných poznatků navrhněte a implementujte systém, který dokáže vyhledat na záznamu kamery konkrétní osobu.
4. Vyhodnoťte výsledky systému na reprezentativním vzorku dat.
5. Vytvořte stručný plakát prezentující práci, její cíle a výsledky.

Literatura:

- dle doporučení vedoucího

Při obhajobě semestrální části projektu je požadováno:

- Funkční prototyp řešení

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci dřívějších projektů (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Smrž Pavel, doc. RNDr., Ph.D.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2017
Datum odevzdání: 23. května 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Abstrakt

Tato práce se zabývá návrhem a implementací systému, který umožňuje vyhledávání a rozpoznávání osob ve videozáznamech. Nejprve je proveden teoretický rozbor související problematiky, poté je na základě tohoto rozboru proveden návrh a dle návrhu dále implementován systém. Pro rozpoznávání obličejů je využito konvolučních neuronových sítí a v implementaci především knihoven dlib a OpenFace. V systému je využito paralelizace a distribuce úkonů mezi více zařízeními a je také dbáno na způsob praktického využití, při kterém můžeme mít často k dispozici jen málo informací o vyhledávané osobě. Systém dosahuje celkové přesnosti detekce a rozpoznávání osob cca 70% až 80% dle úlohy. Lze jej využít např. k vyhledání konkrétní osoby v záznamu, odhadu počtu celkových průchodů či průchodů jedné osoby a také ke zjštění výskytů neznámých osob v daném prostředí.

Abstract

This thesis deals with the design and implementation of a system, which allows to search for and recognize people in video recordings. The presented design is based on a preceding research in theory relating to the topics of face and people recognition. Furthermore, the system design is implemented using convolutional neural networks for face recognition, while the implementation primarily utilizes the libraries dlib and OpenFace. The design and implementation use parallelization and distribution of tasks among multiple devices to reduce computation time, while also bearing in mind the practical applications of such system, such as working with limited amounts of available information regarding the person we seek. The precision of people detection and recognition of the implemented system is about 70% to 80%, based on the performed task. Among other uses, the system can be utilized to find a particular person in a video recording, to estimate the number of passes through the monitored space of one person, or the number of passes in total, or to find unknown people in the monitored space.

Klíčová slova

rozpoznávání obličejů, strojové učení, neuronové sítě, zpracování obrazu, monitorování, vyhledávání osob, paralelizace

Keywords

face recognition, machine learning, neural networks, image processing, monitoring, people search, parallelization

Citace

JEZERSKÝ, Matouš. *Vyhledávání konkrétních osob na záznamech z bezpečnostních kamer.* Brno, 2018. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Doc. RNDr. Pavel Smrž, Ph.D.

Vyhledávání konkrétních osob na záznamech z bezpečnostních kamer

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Doc. RNDr. Pavla Smrže, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Matouš Jezerský
21. května 2018

Poděkování

Chtěl bych poděkovat vedoucímu mé diplomové práce Doc. RNDr. Pavlu Smržovi, Ph.D. za jeho rady a pomoc při tvorbě této práce.

Obsah

Kapitola 1

Úvod

Strojová identifikace a rozpoznávání osob v obrazu jsou velmi aktuálními a rychle se vyvíjejícími tématy, úzce souvisejícími také s bezpečností. Tato práce se zabývá problematikou rozpoznávání a vyhledávání konkrétních osob v záznamech videokamer. Jejím cílem je konkrétně vytvoření systému, který je schopen v záznamech identifikovat osoby, které se v nich vyskytují, a tyto osoby také v jednom či více záznamech vyhledávat na základě jejich referenčních snímků. Systém by měl poskytovat informace ke zjištění počtu vyskytujících se osob na záznamech a také umožňovat určit čas a dobu výskytu osoby v daném videozáznamu. Takovýto systém je pak možné použít např. pro nalezení osob, jejichž výskyt v monitorovaných prostorech je nežádoucí, nebo také k vyhledávání výskytů konkrétní zájmové osoby ve velkém množství záznamů.

Podobná problematika byla také tématem diplomových prací *Sledování a rozpoznávání lidí na videu* [1] a *Konvoluční neuronové sítě pro bezpečnostní aplikace* [2]. Zde je ale volen mírně odlišný přístup. Cílem je vytvořit takový systém, který by umožňoval rychlejší zpracovávání s využitím paralelizace a distribuce zpracovávání mezi více zařízení, a to především při využití výpočetně náročnějších metod detekce a rozpoznávání. Systém také musí poskytovat možnost rychlého vyhledávání v objemném množství dat, jelikož úložiště videozáznamů mohou být velmi rozsáhlá. V této práci je problém řešen tak, že se k videozáznamům vytvářejí soubory s potřebným, avšak výrazně redukovaným množstvím informací, nad kterými se pak provádějí další operace. Tento proces umožňuje vyhnutí se náročnému opětovnému zpracovávání záznamů, a také ušetření úložného prostoru. V návrhu a implementaci programu je také dbáno na praktickou využitelnost. Chceme-li v záznamech vyhledávat např. zloděje, či osobu, o které nemáme velké množství informací, lze předpokládat, že nemáme k dispozici více než jeden či dva různé snímky obličeje této osoby, a systém musí být použitelný i v těchto případech.

Ze začátku, v kapitole 2, se práce věnuje rozboru problematiky související s rozpoznáváním a identifikací osob ve videozáznamech. Hlavním zaměřením je rozpoznávání pomocí obličejů, avšak krátce jsou zmíněny i jiné alternativy. Je uveden teoretický základ k neuronovým sítím a jejich využití v této oblasti, a také popsán celý proces detekce a rozpoznávání. V kapitole 3 je pak uveden návrh celého systému a také je zde popsána jeho implementace. V kapitole 4, jež následuje, je pak uvedena analýza systému a experimentů s ním. Na závěr této kapitoly je uvedeno vyhodnocení na sadě bezpečnostních záznamů.

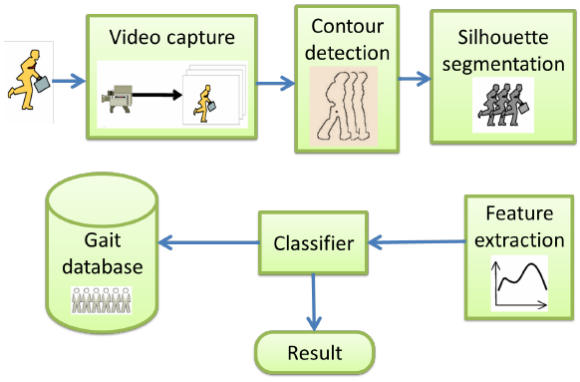
Kapitola 2

Rozbor problematiky

2.1 Detekce a rozpoznávání osob

Hlavním prvkem této práce je právě rozpoznávání jednotlivých osob. Máme-li k dispozici video, tedy sekvenci snímků, existuje mnoho faktorů, které rozpoznávání osob mohou ovlivnit, ale také mnoho forem rozpoznávání a identifikace. Z obrazu můžeme osoby rozpoznávat podle postavy, chůze nebo obličeje. Na úvod kapitoly jsou nejprve krátce zmíněny metody rozpoznávání osob dle pohybu (chůze) a obličeje, avšak detailněni je popisováno pouze rozpoznávání dle obličeje, na které se tato práce zaměřuje.

Při procesu rozpoznávání osob dle pohybu se nejprve provádí segmentace záznamu a detekce pohybu, tímto se samotná osoba izoluje od prostředí/pozadí. Následně se provede detekce kontur, tedy nalezení „ohraničení“ lidského těla v obrazu, a na závěr se provede extrakce rysů z těchto detekovaných kontur a jejich pohybu v sekvenci snímků. Stejně jako jakákoliv jiná detekce a rozpoznávání z obrazu je i tato metoda citlivá na úhel kamery, osvětlení a další faktory související s variabilitou videozáznamů. Jedním z hlavních problému je však proměnlivost samotné chůze osoby. Na styl chůze a pohybu má vliv únava, možná zranění, nebo také i to, že osoba pouze někam spěchá. Všechny tyto faktory pak výrazně komplikují správné rozpoznávání, a tudíž mají negativní dopad na jeho spolehlivost [20]. Systémy využívající rozpoznávání dle chůze vyžadují jistou znalost prostředí (např. znalost pozadí kamerou zabíraného prostoru, pro přesné odlišení procházejících osob od složitějšího, případně proměnlivého pozadí), alternativně i využití jiných detekčních komponent než jsou kamery [26].



Obrázek 2.1: Proces klasifikace osob podle chůze. (Převzato z [21]).

Chceme-li však identifikovat osobu my sami, jednoduchou a jistou metodou (vyjma zvukových vjemů) je pro nás rozpoznání obličeje. Stejně tak se této vlastnosti využívá při strojovém rozpoznávání osob. Obličej je pak možné přiřadit i k dalším, méně rozlišujícím vlastnostem osoby na záznamu. Např. při samotné detekci osoby můžeme v záznamu detekovat postavu, a tuto postavu sledovat, aniž bychom měli povědomí o její identitě, zatímco identifikaci provádí algoritmus pro rozpoznávání obličeje. Nachází-li se pak obličej v hranicích detekované postavy, můžeme tyto dvě informace sloučit, a nadále sledovat pouze postavu i při nedostupnosti obličeje (osoba se otočí, skloní, zakryje si obličej).

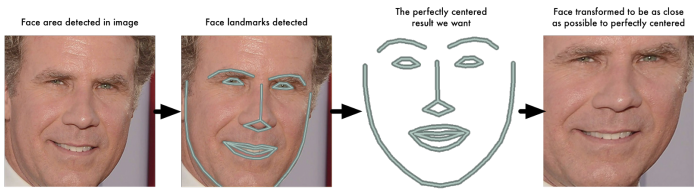
2.2 Detekce a rozpoznávání obličejů

Detekce a rozpoznávání obličejů má z několika fází, lišící se počtem a vlastnostmi dle použitého algoritmu. Jak je zmíněno v Oxfordské publikaci Deep Face Recognition [35], existuje mnoho metod, které tento problém řeší; např. metody, které získávají reprezentaci obrazu na základě lokálních obrazových deskriptorů jako jsou SIFT, LBP či HOG, které jsou následně agregovány do celkového deskriptoru obličeje, a to pomocí metod jako je např. Fisher Vector [34]. Velmi populárními metodami je nyní rozpoznávání obličejů pomocí hlubokých konvolučních neuronových sítí, a to především díky velmi příznivým výsledkům a při vhodné volbě datové sady i invarianci vůči osvětlení a natočení obličeje. To, že řešení pomocí hlubokého hlubokého učení je jak preferovaným a populárním přístupem, tak jednou z nejefektivnějších dostupných metod, se lze přesvědčit nejen ve vyhodnoceních v literatuře, jež se váže k sekci této práce zmiňující hluboké neuronové sítě, ale také díky výsledkům soutěže ImageNet Large Scale Visual Recognition Challenge [38] (především v posledních letech [3]).

Samotnému průchodu neuronovou předchází předzpracování, které síti poskytuje data v ucelené a standardizované formě [35]. Celý proces začíná detekcí obličeje v obrazu. Nejprve je nutné nalézt, kde v obrazu se obličej, nebo obličeje nacházejí (pro zjednodušení je zde uveden proces pro jeden obličej, pro více obličejů je však postup analogický).

Po nalezení (detekci) obličeje se obličej ořízne, a dále jsou v ořízntém obrazu vyhledávány charakteristické body obličeje (facial landmarks/keypoints). Určení těchto bodů se liší dle použitého algoritmu, jedná se však zpravidla o body ohraničující oči, nos, ústa a hranice obličeje.

S pomocí těchto bodů lze vypočítat natočení obličeje, dle kterého pak lze celý obličej transformovat tak, aby body odpovídaly natočení obličeje kolmo vůči kameře. Takto transformovaný obraz je potom odeslán na průchod neuronovou sítí, která z něj extrahuje rysy (embeddings).



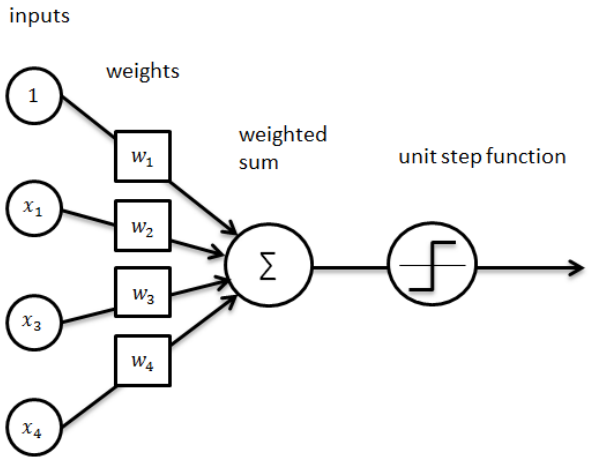
Obrázek 2.2: Příklad předzpracování obličeje (Převzato z [4]).

Tyto rysy jsou vektory hodnot, jejichž význam je sice obtížné (ne-li téměř nemožné) určit, avšak jsou charakteristické pro každou tvář. A nejen to, tyto rysy se také liší mezi odlišnými obličeji více než mezi obličeji stejnými.

Porovnání, tedy identifikace či verifikace obličeje pak může být prováděna výpočtem vzdálenosti mezi vektory (neplatí pro všechny vektorové reprezentace, jelikož to, že se některý prvek mezi vektory velmi liší ještě neznamená, že je relevantní pro skutečnou diferenciaci obličejů), nebo natrénováním klasifikátoru, který se naučí obličeje dle těchto rysů odlišovat.

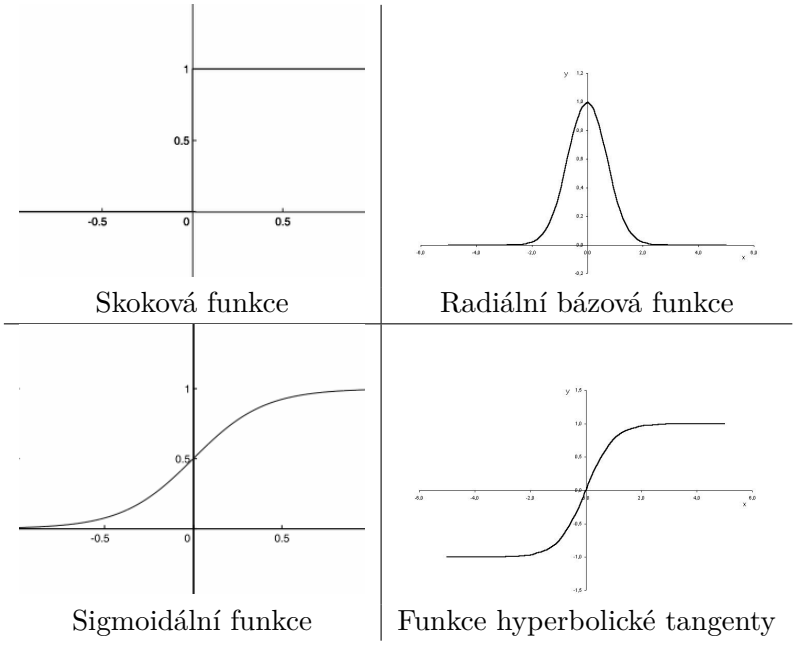
2.3 Umělé neuronové sítě

Umělé neuronové sítě jsou v oblasti soft computingu využívány již více než 50 let. Základní umělý neuron, tzv. perceptron, byl navržen Frankem Rosenblattem [37] a poprvé simulován v roce 1957 v Cornell Aeronautical Laboratory. Nyní je již běžné, že je využíváno jiných modelů než perceptronů, které jsou však stále vhodné pro jednoduchou ilustraci funkce neuronových sítí a stále se tedy využívají k jejich vysvětlení. Perceptron, tedy velmi základní a jednoduchý neuron, vypadá následovně:



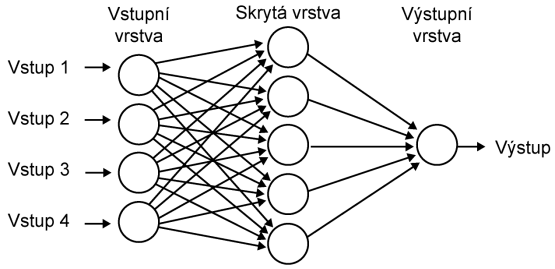
Obrázek 2.3: Perceptron (Převzato z [5]).

Perceptron má množinu vstupů x_i a výstup, resp. výstupy (skutečný výstup je jen jeden, veškerá výstupní propojení nesou stejnou hodnotu). Každý vstup má pak přiřazenou váhu w_i, která slouží jako faktor významu vstupu. Výstup je pak roven výsledku aktivační funkce, kterou je v nejjednodušším modelu perceptronu skoková funkce (podobná funkci signum, pro záporné vstupní hodnoty má však výstupní hodnotu 0), jejíž vstupem je suma všech vstupů, vynásobená svými váhami, tedy $\sum_i x_i w_i$. Tuto sumu, resp. funkci, která tuto sumaci provádí, nazýváme propagační funkcí. Příklady běžných aktivačních funkcí, jako je skoková funkce a další, lze vidět na níže uvedených obrázcích.



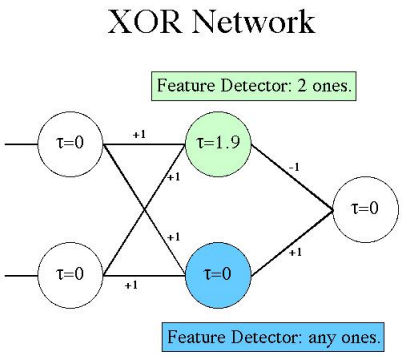
Obrázek 2.4: Příklady aktivačních/přenosových funkcí.

Jednotlivé neurony jsou poté uspořádány do vrstev a propojeny tak, že výstupy některých neuronů jsou zároveň vstupy jiných, případně jsou tyto výstupy zároveň vstupy neuronů samotných. Tyto propojené neurony pak tvoří orientovaný, ohodnocený graf – neuronovou síť.



Obrázek 2.5: Příklad uspořádání neuronů do vrstev (Převzato z [6]).

Spojení v síti přenáší výstupní hodnoty neuronu i do vstupu neuronu j. Neuron i je pak předchůdcem neuronu j, analogicky pak j je následovníkem i. Ke každému spojení se pak váže váha w_{ij}, a hodnoty těchto vah se během učení, modifikují, spolu s prahem v případě některých aktivačních funkcí, aby bylo dosaženo požadovaného výstupu sítě.
Neuronová síť, která je uvedená na obrázku 2.5, obsahuje tři vrstvy: vstupní, skrytou a výstupní. Takováto neuronová síť je již schopna aproximovat i složitější funkce. Např. na rozdíl od jednovrstvé sítě, je vícevrstvá síť schopná se naučit provádět funkci XOR. Příklad takovéto neuronové sítě lze vidět na obrázku 2.6.



Obrázek 2.6: Neuronová síť vykonávající funkci XOR (Převzato z [7]).

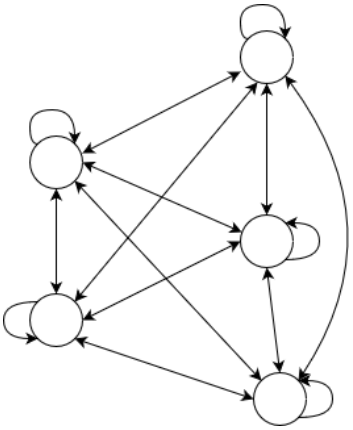
V této konkrétní síti lze vidět, že neurony skryté vrstvy vykonávají funkci rozpoznávání výskytu „jedniček“, tedy binárních hodnot 1. Zeleně označený neuron tedy funguje jako AND a modrý neuron jako OR. Váhy jsou naznačeny číselně u spojnic neuronů, práh pro aktivaci je značen hodnotou τ. Neuron výstupní vrstvy je pak aktivován, pokud vstup obsahuje právě jednu hodnotu 1. Pokud se vyskytují dvě hodnoty jedna, aktivuje se zeleně označený neuron, ovšem váha jeho výstupu vůči vstupu posledního neuronu je záporná, poslední neuron se tedy neaktivuje. Pokud naopak na vstupu žádná hodnota 1 není, poslední (výstupní) neuron dostává na vstup celkově 0, a k jeho aktivaci opět nedojde. Variant, jak mohou být určeny váhy a prahy aktivačních funkcí, aby síť prováděla funkci XOR je mnoho, a jsou ustanoveny během fáze učení.

2.3.1 Základní klasifikace sítí

V dnešní době existuje velké množství neuronových sítí – perceptrony, MLP, rekurentní sítě, konvoluční sítě, autoenkodéry a mnoho dalších. Přes velmi vysokou variabilitu, která je dána především vhodností jistých typů sítí pro určité problémy, např. LSTM sítě pro predikci sekvencí [25], konvoluční sítě pro zpracování obrazu [27] apod., níže uvedu základní klasifikaci sítí dle architektury či topologie a také dle typu učení sítě. Níže uvedená klasifikace a vysvětlení tříd se řídí dle knihy [32].

Dle architektury klasifikujeme sítě následovně:

- Plně propojená síť
  Všechny uzly této sítě jsou vzájemně propojeny. Tyto spojení mohou být excitační (pozitivní váhy), inhibiční (negativní váhy) či irelevantní (téměř nulové váhy). Jedná se o nejobecnější architekturu neuronové sítě, ostatní třídy tedy mohou být považovány za její speciální případy. Vynechání spojení je totiž ekvivalentní s nastavením váhy tohoto spojení na nulu.
  Tento typ dále dělíme na symetrické plně propojené sítě (pro všechny spojení mezi libovolnými neurony i a j platí, že $w_{ij} = w_{ji}$) a asymetrické (w_{ij} a w_{ji} se mohou lišit). Tato architektura je používána jen zřídka, a to kvůli velkému množství parametrů pro učení. Síť s n neurony má totiž n^2 vah.

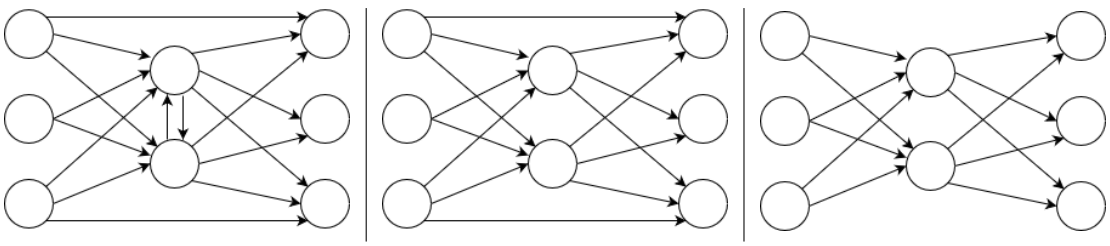


Obrázek 2.7: Plně propojená neuronová síť.

- Vrstvená síť
  Uzly těchto neuronových sítí jsou rozděleny do podmnožin zvaných vrstvy takovým způsobem, že žádná spojení z vrstvy j nevedou do vrstvy i, pokud vrstva j následuje vrstvu i. Konvencí je nazývat první vrstvu, tedy vrstvu číslo 0, vrstvou vstupní (input layer). V této vstupní vrstvě pak nedochází k žádným výpočtům, pouze k distribuci hodnot do dalších vrstev. Ve vstupní vrstvě také nejsou žádná vnitřní spojení (neurony vrstvy nejsou vzájemně propojeny). V ostatních vrstvách mohou pak existovat vnitřní spojení a stejně tak mohou být neurony vrstvy i propojeny s neurony vrstvy j, pokud i předchází j, bez nutnosti bezprostředního následnictví vrstvy j.

- Acyklická síť
  Acyklické sítě jsou podtřídou vrstvených sítí, které neobsahují žádné spojení uvnitř samotných vrstev, tedy spojení muže existovat mezi vrstvami i a j, pokud i předchází j a zároveň $i \neq j$. Výpočetní proces těchto sítí je pak výrazně jednodušší, něž u cyklických či vnitřně propojených vrstvených sítí. Sítě, které nejsou acyklické, také nazývají **rekurentní**.

- Dopředná síť
  Dopředné sítě jsou podtřídou acyklických sítí, ve kterých je povoleno propojení pouze přímo sousedících vrstev, a to opět tak, že vedou z vrstvy i do vrstvy j, pokud j následuje bezprostředně po i. Dopředné sítě patří mezi nejběžněji používané neuronové sítě. Koncept těchto sítí je takový, že každá následující vrstva abstrahuje rysy na vyšší úrovni abstrakce z vrstev předcházejících.



Obrázek 2.8: Vrstvená, acyklická a dopředná neuronová síť.

Jak bylo již výše zmíněno, neuronové sítě aproximují požadovanou funkci po fázi učení, tedy po úpravě a adaptaci hodnot vah vstupů a prahů aktivačních funkcí dle tzv. trénovacích dat.
Učení neuronových sítí se pak dělí na:

- Korelační učení
  Korelační učení je jedním z nejdéle známých principů biologického učení, popsáno D. O. Hebbem [29], někdy také nazývané Hebbovské učení. Takové učení se řídí stejnojmenným Hebbovým principem, z něhož vyplývá, že jsou-li dva neurony současně aktivní, měly by mít vyšší stupeň vzájemné interakce než neurony, jejichž aktivita korelaci nevykazuje. V takovém případě by jejich vzájemná interakce měla být buď nulová, nebo velmi malá [8]. Jinými slovy se váhy mezi dvěma aktivními neurony posilují, opačně pak zeslabují, např. tedy vede-li aktivace nějakého neuronu k aktivaci požadovaného výstupního neuronu, dojde k posílení vah na jejich spojení. Síla těchto spojení pak eventuálně odpovídá korelaci mezi vstupy a výstupy.

- Soutěživé učení
  Tento typ učení funguje na odlišném principu, a to pokud je vstupní vzor přiveden na vstup neuronové sítě, odlišné uzly soutěží o nejvyšší úroveň aktivity pomocí excitace sama sebe a inhibice mezi ostatními uzly. Poté dojde k upravení vah mezi vstupními uzly a vítězným uzlem tak, že dojde ke zvýšení pravděpodobnosti opětovného vítězství tohoto uzlu ve fázi soutěžení pro podobné vstupní vzory. Takto postupem času dojde k specializaci každého uzlu na určitý typ podobných vstupních vzorů.

- Zpětnovazebné adaptivní učení
  Zpětnovazebné učení se zaměřuje na ladění parametrů na základě odezvy. Pokud tedy např. posílení určité váhy vede ke zhoršení výsledku, pak je tato váha namísto toho snížena. Tyto prováděné změny jsou zpravidla velmi malé, a to z důvodů méně prudkých výchylek, což vede k lepší toleranci v chybách učitele, odezvy, či mechanizmu, jež vyhodnocuje výsledek sítě. Velikost této změny ovšem nesmí být příliš malá, jelikož by tyto drobné kroky vedly k velmi dlouhým časům potřebným k učení sítě. Některé učící algoritmy pak velikost této změny průběžně mění.

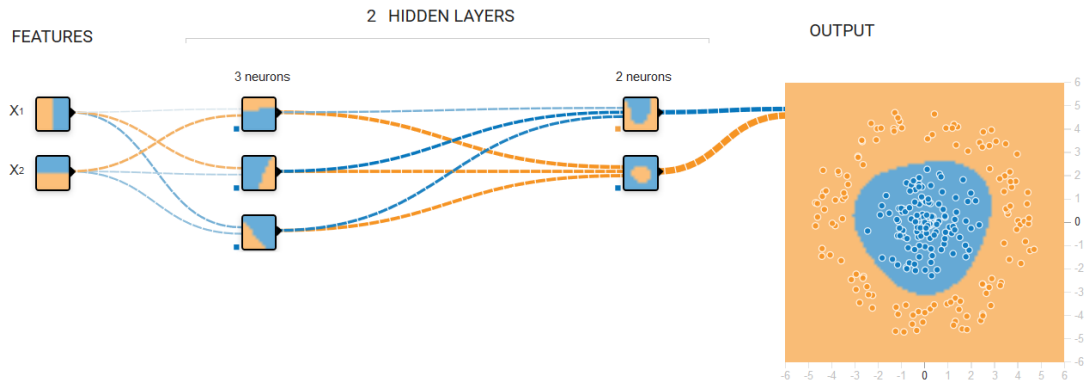
2.4 Učení neuronových sítí

Parametry neuronových sítí jsou modifikovány pro aproximaci požadované funkce tzv. učícími algoritmy. Strojové učení, tedy modifikace parametrů na základě poskytnutých dat, definuje Mitchell [9] následovně: „Počítačový program je schopný učit se ze zkušeností E pro vykonávání určité třídy úloh T a metriky výkonnosti P, pokud se tato metrika zlepšuje ve vykonávání třídy úloh T se získáváním zkušeností E.“

T je třída úloh či problémů, do níž spadá úloha, kterou program řeší. V kontextu neuronových sítí tedy třída funkcí, do niž spadá funkce, jež neuronovou sítí aproximujeme. Je to tedy např. klasifikace, tedy zařazení vstupu do určité třídy či tříd, regrese neboli předpovídání číselné hodnoty v závislosti na vstupu, či třeba překlad textu. V knize Deep Learning [27] je třída úloh T popisována jako způsob, kterým systém strojového učení zpracovává příklad. Příklad pak definuje jako soubor rysů, jež byly kvantitativně měřeny na nějakém objektu nebo události, jež chceme systémem zpracovat. Tento příklad pak reprezentujeme vektorem $X \in \mathbb{R}^n$, kde každý prvek x_i z tohoto vektoru je odlišným rysem, např. rysy obrazových dat jsou většinou hodnoty pixelů. Zmiňovaným příkladem je pak vektor těchto

rysů, tedy samotný obraz. Tříd úloh T se v literatuře nachází mnoho, a různé zdroje uvádí odlišné klasifikace a rozdělení, uvedu tedy pouze některé z nich, a to především ty, které přímo souvisí s problematikou řešenou v této práci.

- **Klasifikace** je typ úlohy, kde se program snaží zařadit vstup do jedné z k kategorií. Při učení se tedy snažíme, aby neuronová síť prováděla zobrazení f vstupu x do množiny tříd $\{1, \ldots, k\}$. Existují i odlišné typy klasifikace, např. takové, kde výstupem není pouze jedna předpokládaná třída, ale rozložení pravděpodobnosti náležitosti vstupu do množiny tříd. V této práci je klasifikace využitelná pro identifikaci osoby na základě rysů obličeje, v jiných případech se může jednat o odhadnutí ručně psaného písmene či čísla, nebo typu objektu v obraze. Typ neuronové sítě, její architektura, velikost a použité algoritmy se liší dle klasifikačního problému. Pro klasifikaci objektů v obraze mohou být potřeba hluboké konvoluční sítě, avšak pro klasifikaci číslic, např. z datové sady MNIST, často využívané pro demonstrační či výukové účely [10], mohou stačit velmi malé a jednoduché dopředné sítě. Příkladem jednoduché sítě provádějící úlohu klasifikace je níže uvedený obrázek.



Obrázek 2.9: Příklad neuronové sítě pro klasifikaci a její reakce na vstup. Body v grafickém zobrazení výstupu znázorňují trénovací vstup, barva na pozadí pak třídy, do kterých by vstupní body spadaly, vyskytovaly-li by se v dané oblasti. Váhy spojení jsou znázorněny tloušťkou spojující čáry barevně odlišující polaritu hodnoty. U každého neuronu je také graficky znázorněno, jakou funkci vykonává. (Vytvořeno ve webovém nástroji TensorFlow [11].

- **Úlohy se strukturovaným výstupem** jsou takové úlohy, kde je výstupem neuronové sítě strukturovaná reprezentace vstupu, velmi často ve formě vektoru. V případě zpracování obrazu se tak jedná a zobrazení jednoho vektoru na jiný. V kontextu této práce je tato úloha prováděna při extrakci rysů obličeje – vstupní obrázek ve vektorové reprezentaci projde neuronovou sítí, a jejím výstupem je pak vektor rysů. Výstupní struktura, právě jako v tomto případě, nemusí nutně být člověkem lehce interpretovatelná, záleží především na charakteristice prvků této struktury, jejichž účelem je rozlišovat třídy vstupu. Při extrakci rysů obličeje pak vyžadujeme, aby jisté prvky výstupního vektoru určovaly míru podobnosti více obličejů, tedy poskytovaly informace k identifikaci obličeje.

Ke kvantifikaci schopnosti neuronové sítě vykonávat danou úlohu pak využíváme metriku výkonnosti P. Tato metrika se nemusí nutně vztahovat právě k vykonávané úloze, avšak ve většině případů tomu tak je [27]. Tuto metriku však nevyhodnocujeme na stejných datech, na kterých neuronovou síť trénujeme, nýbrž na tzv. testovací sadě. Tato sada je oddělena od dat trénovacích, jelikož při ověřování schopnosti výkonu úlohy na samotných trénovacích datech by docházelo ke zkreslení požadovaného výsledku přílišnou specifikací sítě právě na těchto datech. V případě vyhodnocování metriky na trénovacích datech by totiž při přeučení došlo k velmi dobrým výsledkům právě na této sadě, avšak na neznámých datech by výkonnost byla mizivá, přestože by metrika P vykazovala velkou úspěšnost, resp. nízkou chybovost.

Pro úlohy jako je klasifikace je metrika P často volena jako přesnost modelu. Přesnost či správnost (accuracy) je pak vypočítána na základě poměru příkladů, tedy jednotlivých vstupů, v jejichž případě dochází ke správné klasifikaci, oproti vstupům klasifikovaným nesprávně. Podobně určujeme také chybovost, která je inverzní hodnotou k přesnosti.

Správnost může být však velmi často zavádějící. Máme-li data, v nichž je jedna ze tříd reprezentována velmi malým počtem vzorků, např. při hledání nálezů ve snímcích sítnice, lze předpokládat, že velké množství snímků žádný nález obsahovat nebude. Pokud bychom pak měli 1 snímek s nálezem a 99 bez nálezu a neuronová síť by všechny vyhodnotila jako bez nálezu, samotná správnost by byla 99%, což není vypovídající. Používáme tedy hodnoty přesnost či preciznost (precision) a úplnost (recall). Nadále bude v tomto textu preciznost uváděna jako přesnost, termín přesnost tedy bude reprezentovat „precision“ nikoliv „accuracy“. Tyto hodnoty lze odvodit z tzv. matice záměn (confusion matrix) následovně:

		Předvídané	
		Pozitivní	Negativní
Skutečné	Pozitivní	TP	FN
	Negativní	FP	TN

Tabulka 2.1: Matice záměn.

$$\text{správnost} = \frac{TP+TN}{TP+FN+FP+TN}$$
$$\text{přesnost} = \frac{TP}{TP+FP}$$
$$\text{úplnost} = \frac{TP}{TP+FN}$$

V uvedeném příkladu by pak přesnost i úplnost byly 0%, což nám již říká, na rozdíl od 99% správnosti, že klasifikace nefunguje tak, jak bychom chtěli.

K vysvětlení definice strojového učení pak zbývá popsat zkušenost E. Kniha Deep Learning [27] dělí zkušenosti E na základě typu učení neuronové sítě. Toto rozlišení odpovídá typu učení s učitelem (supervised learning) a bez učitele (unsupervised learning), jelikož právě tyto typy určují, v jaké formě neuronová síť zkušenosti přijímá.

Učící algoritmy bez učitele získávají zkušenosti z datové sady vstupů s různými vlastnostmi a rysy, které se nadále snaží síť naučit a rozlišovat. Jedná se tedy o odvození struktury z neoznačených dat. Úlohy využívající učení bez učitele jsou zahrnuty v Hebbovském učení, avšak vyžadují odlišný přístup než úlohy využívající učení s učitelem, jako je např. klasifikace. U tohoto přístupu je totiž nutné volit odlišné metriky, jelikož správnost či přesnost nelze určit – chybí totiž porovnání s označenými daty. Měření úspěšnosti a výkonnosti

sítě je tak výrazně obtížnější. Např. u shlukování (clustering) se analyzuje podobnost entit ve shluku a jejich odlišnost od entit v ostatních shlucích [12].

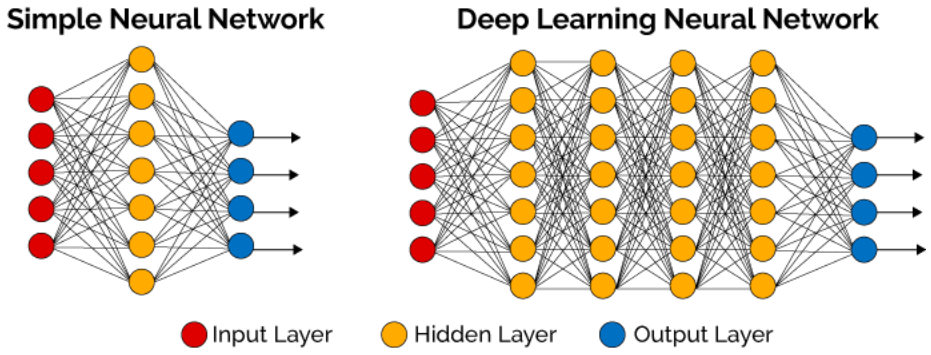
Učící algoritmy s učitelem získávají zkušenosti z datové sady vstupů, kde každý vstup je zároveň označen. Jedna taková značka (label) pak odpovídá správnému výstupu, jež je od neuronové sítě očekáván, dostane-li na vstup takto označená data. Např. vektor hodnot pixelů reprezentující obrázek obsahující ručně napsanou číslici 3 bude označen značkou „3“. Při učení, testování a validaci pak víme, jaký výstup je očekáván, a můžeme určit správnost odezvy sítě.

Tyto typy učení se však týkají algoritmů, které využívají datových sad, jež jsou v průběhu učení neměnné. Opačným příkladem je pak posilované či zpětnovazebné učení (reinforcement learning), kde je součástí zkušenosti E také prostředí, nejen fixní datová sada. Toto prostředí je algoritmem ovlivňováno, a dochází i ke zpětnému přenosu informace z prostředí zpět k algoritmu, tedy ke zpětné vazbě, jež je v procesu učení využívána.

2.5 Hluboké učení (Deep Learning)

Tato podkapitola se zabývá hlubokými neuronovými sítěmi (deep neural networks) a hlubokým učením (deep learning). Jelikož rozsah této problematiky je velmi široký, a to i v případě soustředění se na části, jež jsou k této práci relevantní, zaměřuji se v následujících podkapitolách a sekcích především na zásadní informace související s řešenou problematikou. Velká část informací z této podkapitoly je získána z knihy Deep Learning [27] od autorů Ian Goodfellow, Yoshua Bengio a Aaron Courville.

Ze začátku stručně uvedu zásadní rozdíl mezi běžnými a hlubokými neuronovými sítěmi. Běžná síť na rozdíl od hluboké může obsahovat pouze jednu nebo dvě skryté vrstvy. Zpravidla je tomu tak, že pokud má neuronová síť více než dvě skryté vrstvy, považujeme ji za hlubokou (viz obr. 2.10). Běžná neuronová síť je tedy méně komplexní, a pro stejný úkol vyžaduje více informací o rysech, které je po dostatečném trénování (vhodná a velká datová sada, dostatečně velký model) hluboká síť schopna odvodit sama.



Obrázek 2.10: Porovnání neuronových sítí. (Převzato z [13]).

Existuje mnoho metod a přístupů k hlubokému učení. Ne všechny navrhované přístupy jsou však vhodné pro praktické využití. Mezi pokročilé a ambiciózní přístupy se řadí např. různé metody učení bez učitele, jako jsou strukturované probabilistické modely, hluboké generativní modely, či autoenkodéry – které jsou využívány ve strojovém učení již od roku 1987 (LeCun), avšak nyní začínají být využívány i v generativním modelování.

Metody zmíněné v této práci jsou však široce využívané, ověřené části hlubokého učení, a to jak v teoretických, tak praktických oblastech (průmysl, bezpečnost, biometrie).

Nynější využívané metody hlubokého učení poskytují silný základ pro učení s učitelem. Přidáváním více vrstev a prvků v jednotlivých vrstvách, můžeme zvyšovat komplexitu funkcí, které může hluboká neuronová síť reprezentovat. Pomocí technologií hlubokého učení a neuronových sítí pak můžeme strojově vykonávat úlohy, které jsou velmi obtížně definovatelné matematicky, avšak pro lidské myšlení se jedná a často jednoduché a intuitivní procesy. Takovou úlohou je velmi často zobrazení vektoru na jiný vektor, jako např. převod obrazu obličeje na vektorovou reprezentaci, která obsahuje vztahy klíčových bodů obličeje, ze kterých je možné osobu jednoznačně identifikovat. Tuto úlohu vykonává lidský mozek automaticky, aniž bychom si to uvědomovali [28]. A právě s takovouto úlohou se hluboké neuronové sítě dokáží vypořádat, jsou-li modely dostatečně velké, a mají-li učící algoritmy k dispozici dostatečně rozsáhlé datové sady trénovacích vzorků.

Základní technologie parametrické funkční aproximace, kterou hluboké učení provádí, a která stojí za velkou většinou moderních metod hlubokého učení, je uvedena v následující sekci ve formě hluboké dopředné sítě, která tyto aproximované funkce reprezentuje.

2.5.1 Hluboké dopředné sítě

Hluboké dopředné sítě, často také nazývané dopředné neuronové sítě nebo vícevrstvé perceptrony (MLPs – Multi-Layer Perceptrons), jsou základní modely pro hluboké učení. Cílem takovéto dopředné sítě je aproximovat nějakou funkci $f*$. Vezměme si např. klasifikátor, síť potom provádí klasifikaci $y = f*(x)$, tedy zobrazení vstupu x do kategorie y. Dopředná síť pak obecně definuje zobrazení $y = f(x;\theta)$, a učí se takové hodnoty parametrů θ, které vedou k nejlepší aproximaci hledané funkce.

Tyto modely se pak nazývají dopředné, protože informace, jež je vyhodnocena ze vstupu x prochází skrze výpočetní vrstvy použité k definování funkce f až do výstupu y. Jak je již v základu popsáno v sekci 2.3, síť neobsahuje žádné zpětné vazby, které by výstupy modelu do něj zasílaly zpět. Jsou-li dopředné sítě rozšířeny o zpětné vazby, stávají se rekurentními.

Hluboké neuronové sítě, na rozdíl od lineárních modelů, umožňují aproximaci nejen lineárních, ale i nelineárních funkcí. Abychom rozšířili lineární modely o schopnost reprezentovat nelineární funkce pro x, můžeme aplikovat lineární model ne na vstup x samotný, ale na jeho transformovanou formu $\phi(x)$, kde ϕ je nelineární transformace. Transformaci ϕ si potom můžeme představit jako funkci poskytující množinu rysů reprezentujících x, nebo jako funkci poskytující novou reprezentaci vstupu x.

Poté je ovšem třeba řešit, jakým způsobem zvolit zobrazení ϕ. V předchozích metodách bylo možné zvolit dostatečně veliký obor hodnot ϕ tak, aby funkce zvládla pokrýt celou trénovací sadu. Ovšem v takových případech bývá generalizace provedená funkcí velmi slabá, stejně tak, jako výsledky na sadě testovací. Další možností je ϕ zvolit ručně, dle pečlivě vybraných vlastností a rysů vstupu. Např. víme-li, že důležitým prvkem, jenž rozlišuje jednotlivá slova v lidském hlasu, je hlasitost, můžeme tohoto faktu využít při ručním návrhu ϕ.

Hluboké učení se však s tímto problémem potýká jiným způsobem. Volenou strategií je se ϕ naučit, tedy odvodit jej ze závislostí vstupů a výstupů. V tomto přístupu máme model $y = f(x;\theta,w) = \phi(x;\theta)^\top w$, kde θ jsou parametry použité pro naučení ϕ z širokého rozsahu funkcí, a také parametry w, které zobrazují $\phi(x)$ na požadovaný výstup. V tomto případě se jedná o příklad hluboké dopředné sítě, kde ϕ definuje skrytou vrstvu. V tomto přístupu

pak parametrizujeme tuto reprezentaci jako $\phi(x;\theta)$ a používáme optimalizační algoritmus k nalezení takového θ, které koresponduje s co možná nejvhodnější reprezentací.

Tento postup pak lze ještě upravit podobným způsobem, jako ten, který byl již předem zmíněn – tedy je možné jisté části ručně upravit, aby došlo ke zlepšení, či rychlejšímu nalezení požadované reprezentace ϕ. Konkrétně tedy tak, že lze využít znalostí problému k vymezení třídy funkcí, ze kterých ϕ volíme. Velmi obecně totiž můžeme uvažovat téměř všechny nelineární funkce, avšak pokud víme, že k nejlepším výsledkům vedou např. operace jako je konvoluce, můžeme pak třídu funkcí, v níž ϕ hledáme, dle této znalosti vymezit.

2.5.2 Konvoluční sítě

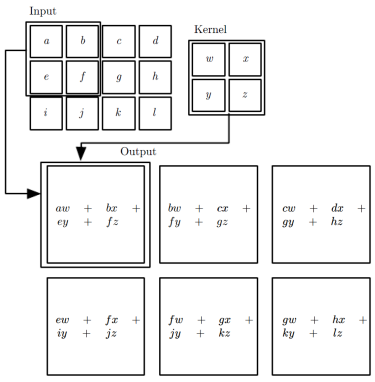
Konvoluční sítě (také konvoluční neuronové sítě, či CNN) jsou specializací neuronových sítí pro zpracovávání dat, které mají známou mřížkovou/maticovou topologii. Příkladem takových dat je třeba časová řada, kterou lze považovat za jednorozměrnou mřížku pravidelných časových vzorků, nebo třeba obrazová data, která jsou reprezentována dvourozměrnou mřížkou pixelů, případně pak vícerozměrnou, uvažujeme-li také barvy pixelu. Konvoluční neuronové sítě jsou velmi úspěšné v praktických aplikacích, jako je zpracování obrazu, zvuku apod. Jak již název napovídá, v těchto sítích je využito speciální lineární operace – konvoluce. Hlavní rozdíl mezi konvolučními sítěmi a těmi běžnými je pak využití operace konvoluce namísto maticového násobení v alespoň jedné z vrstev. Samotná operace konvoluce je pak značená hvězdičkou, a pro spojité hodnoty definována následovně:

$$s(t) = (x * w)(t) = \int_{-\infty}^{\infty} x(a)w(t-a)da$$

Diskrétní konvoluce (používaná v konvolučních sítích) je potom definována jako:

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a)$$

V terminologii konvolučních sítí poté označujeme první argument funkce (x) jako vstup, druhý argument funkce (w) jako jádro, a celkový výstup funkce je nazýván příznaková mapa.



Obrázek 2.11: Příklad dvourozměrné konvoluce. Převzato z knihy Deep Learning [27].

V aplikacích konvolučních sítí je vstupem většinou vícerozměrné pole dat, jádrem pak vícerozměrné pole parametrů, jež jsou adaptovány učícím algoritmem. Ve velkém množství případů je žádoucí, abychom byli schopni zpracovávat více než jednu dimenzi dat, pak

konvoluci používáme na vícerozměrném vstupu a stejně tak i vícerozměrném jádru. Mámeli tedy na vstupu dvourozměrný obraz I, používáme i dvourozměrné jádro K. Konvoluce pak vypadá následovně:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n)$$

Jelikož je operace konvoluce komutativní, lze využít ekvivalentní zápis:

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n)$$

V oblasti konvolučních sítí je také často využívána operace křížové korelace (cross-correlation), která bývá v literatuře a softwarových knihovnách nazývána také konvolucí. Křížová korelace je pro dvourozměrný vstup definována následovně:

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n)$$

Je podstatné podotknout, že samotná operace křížové korelace či konvoluce bývá prakticky často využívána ještě s dalšími přídavnými funkcemi. V jedné vrstvě pak nepoužíváme pouze čistou konvoluci, ale ještě další funkce či operace.

2.6 Datové sady

Datová sada (dataset) je kolekce strukturovaných dat. V kontextu neuronových sítí se pak jedná o množinu vstupů a v případech učení s učitelem také očekávaných výstupů či značek. Příkladem datové sady pak může být kolekce obrázků s informacemi o již klasifikovaných objektech, které se v těchto obrázcích vykytují, nebo např. historie změn určité hodnoty pro předpověď sekvencí apod.

Pro účely tohoto projektu je využito datových sad obsahujících označené a klasifikované obličeje. V literatuře se vyskytuje několik takovýchto sad, ne všechny jsou však veřejně dostupné, nebo jsou dostupné jen na písemnou žádost. Např. sada využívaná ve Facebook Research – Social Face Classification dataset (přes 4 miliony obrázků) [40] či sada Celeb Faces (cca 202 000 obrázků) [39] patří mezi rozšířené a rozsáhlé, avšak pro veřejnost nedostupné sady. V tomto projektu (mimo předtrénované modely) byla využita především datová sada LFW (13 233 obrázků) [30], která je sice menší, avšak vhodná pro porovnání výsledků s jinými výzkumnymi prácemi díky své rozšířenosti. Z důvodu dostupnosti datových sad se tak může kvalita trénovaných modelů výrazně lišit, jelikož hluboké neuronové sítě vyžadují velké množství trénovacích dat, a ne každá veřejná datová sada je tak rozsáhlá, jako některé soukromé.

2.7 Sledování objektů

Během zpracovávání snímků průchodu osoby může dojít k selhání detekce obličeje, přestože se osoba ve snímku, v němž k selhání došlo, vyskytuje. K takovéto situaci dochází např. při otočení či sklonění dané osoby. Pokud běžná detekce selže, lze zvolit možné alternativy pro detekci osoby, a to na základě znalosti výskytu osoby v předchozích snímcích.

Jednou z možností jsou tzv. sledovací algoritmy (tracking algorithms), které bývají v oblasti sledování osob široce využívány [23]. Tyto algoritmy umožňují sledovat daný objekt

v optickém toku, a to pomocí sledování klíčových vlastností objektu. Mezi takové algoritmy patří např. Kanade-Lucas-Tomasi tracker (KLT) [41] či algoritmy sledování pomocí korelace a korespondence, využívající korelační filtry, které umožňují sledovat pohybující se, zmenšující či zvětšující se, nebo rotující objekt. Korelační sledovací algoritmy a KLT jsou dále zmíněny a využity v této práci především z důvodů jejich používání v podobných programech či textech (např. v široce využívané knihovně dlib [31]), a přestože je v práci ověřováno více variant, detaily těchto algoritmů a dalšími alternativami se text nezabývá.

Sledovací algoritmy je také možné použít k urychlení procesu detekce a rozpoznávání ve snímcích videa, jelikož je v určitých snímcích možné vynechat výpočetně náročnější detekci obličejů a nahradit ji pouze sledováním pohybu obličeje z předchozích snímků, kdy detekce byla použita. Takto je možné např. spouštět detektor pouze pro jeden snímek z pěti, pokud si můžeme být jisti, že v případě průchodu osoby je zaručeno, že i v případě takovéhoto vynechávání snímků se nám nestane, že osoba nebude vůbec detekována. Je tedy potřeba nalézt vhodnou rovnováhu mezi urychlením a spolehlivostí detekce.

Kapitola 3

Návrh a implementace

Jedním z hlavních požadavků na implementovaný program je vyhledávání osob ve videozáznamech, což implikuje potřebu osoby rozpoznávat, tedy mít možnost provádět identifikaci a verifikaci, dále pak možnost zjištění počtu výskytů osob na videozáznamech a také sledování pohybu daných osob.

Primárním zaměřením projektu je tedy samostatné rozpoznání osoby v proudu videa, tedy v konkrétních snímcích, případně pak možnost nalezení osoby vymykající se z očekávané množiny osob. Např. člověk, který není uvedený v informačním systému jako zaměstnanec (tedy jeho obličej není v databázi obličejů očekávaných osob), může být označen jako nežádaná osoba při jeho výskytu v prostorech s přístupem omezeným pouze pro zaměstnance. Při zvažování metod detekce a rozpoznávání je třeba dbát na přesnost těchto metod, ale také na jejich rychlost. V případě, že máme nepříliš dlouhý záznam a snažíme se kupříkladu pouze jednou za čas najít podezřelou osobu v hodinovém intervalu očekávaného výskytu, není rychlost příliš zásadní. Máme-li však např. záznam z bezpečnostní kamery na chodbě budovy s omezeným přístupem, a chceme tuto chodbu automaticky monitorovat (nemáme tedy pevně stanovený krátký časový interval), může se stát, že potřebujeme data zpracovávat v reálném čase, nebo zpracovávat velké videosoubory. Např. videosoubor se snímkovou frekvencí 30 FPS o délce jedné hodiny obsahuje 108000 snímků. Pro nastínění důležitosti faktoru rychlosti a formy zpracovávání uvažujme následující příklad: pokud bychom zpracovávali 10 hodinový (30 FPS) záznam rychlostí 2 snímky za sekundu bez vynechávání snímků, proces by zabral 150 hodin.

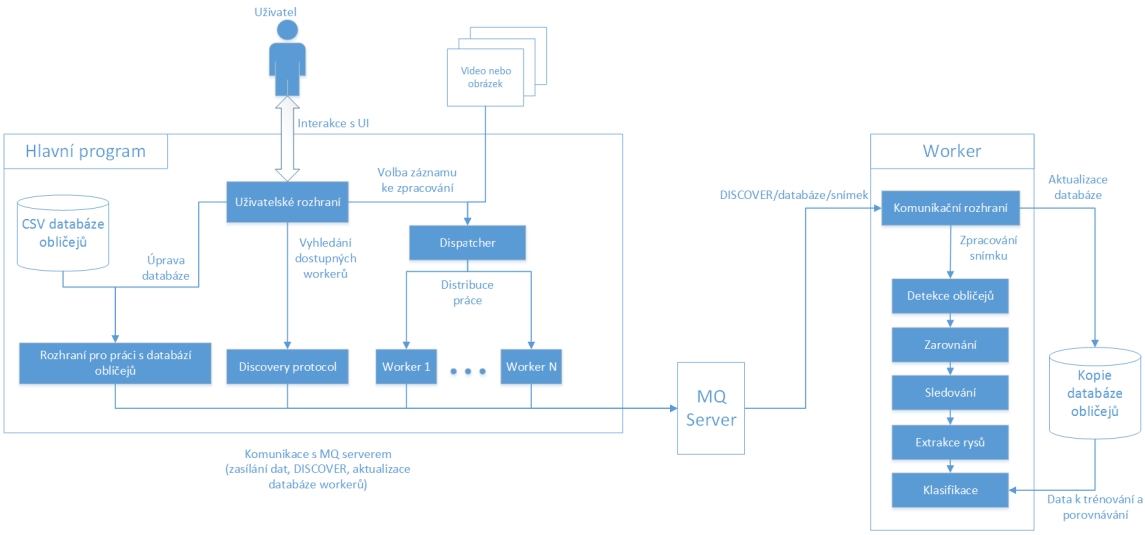
Navrhovaným řešením je tedy program, který poskytuje dvě formy rozhraní. Grafické rozhraní, které umožňuje uživateli zobrazit zpracované videozáznamy a informace o osobách, které se v nich vyskytují, a textové rozhraní, které umožňuje jak vzdálené spouštění programu bez nutnosti grafického prostředí, tak automatizaci spouštění a případnou integraci do jiného systému. Navrhovaný program poskytuje urychlení s využitím paralelizace, kdy se jednotlivé úlohy zpracovávají v jednom či více procesech, na jednom či více zařízeních. Vzhledem k tomu, že je díky paralelizaci možné zpracovávání spouštět na virtuálních instancích, cloudových systémech či výpočetních clusterech, implementace bere ohled na možné rozlišnosti mezi systémy, a program je tedy implementován multiplafromně. Pro možnost jednoduché práce s informacemi, jež jsou získávány z videozáznamů, se tato data ukládají ve speciálních CSV souborech. Tato forma ukládání umožňuje analytické zpracování, zjišťování výskytů osob, opětovnou práci s daty v programu či další zpracovávání i surových dat (detekovaných osob a jejich vektorů rysů), a to bez nutnosti opakovaného, náročného zpracovávání videa. Program také využívá technologie nVidia CUDA [33] a cudNN [14]

pro akceleraci zpracovávání snímků neuronovou sítí s pomocí grafického čipu nVidia, je-li dostupný.

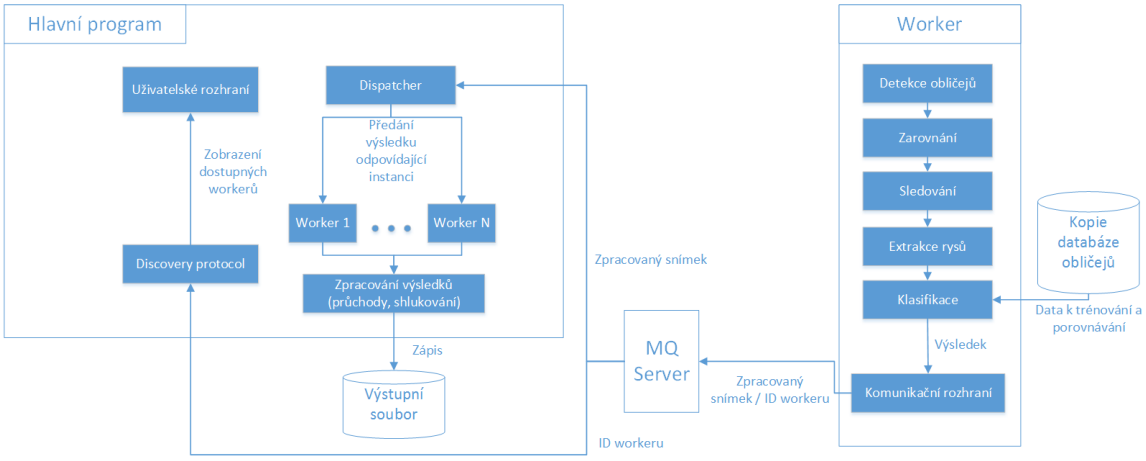
Na základě informací z předchozí kapitoly, byly k řešení problému zvoleny konvoluční neuronové sítě 2.5, a aplikovány další techniky pro urychlení zpracovávání záznamu, jako je využití sledovacích algoritmů a paralelizace. Jelikož je sledování osob komplexním problémem, program byl dekomponován na několik součástí:

1. Hlavní program s uživatelským rozhraním
   - Zpracování vstupního záznamu
   - Databáze osob
   - Distribuce práce mezi procesy pro detekci a rozpoznávání
   - Komunikace s ostatními procesy
   - Zpracování výsledku
   - Zobrazení získaných informací
2. Program poskytující službu detekce a rozpoznávání (dále zmiňován také jako worker)
   - Komunikace s ostatními procesy
   - Detekce obličejů a zarovnání
   - Extrakce rysů
   - Sledování objektů
   - Klasifikace a identifikace
3. Server zprostředkovávající komunikaci mezi procesy (message broker)

Interakci jednotlivých komponent lze vidět na následujících diagramech.



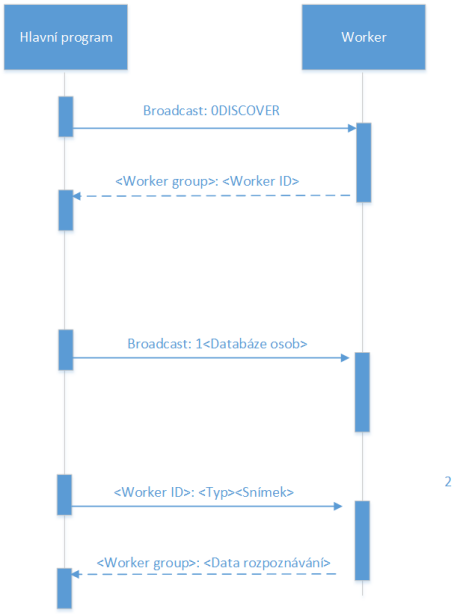
Obrázek 3.1: Diagram návrhu zpracovávání dat a interakce komponent ve směru hlavní program -> worker, tedy zaslání žádosti či dat pro worker z hlavního programu.



Obrázek 3.2: Diagram návrhu zpracovávání dat a interakce komponent ve směru worker -> hlavní program, tedy odpověď workeru na požadavek hlavního programu.

3.1 Komunikační protokol

Pro zpracovávání zpráv byl zvolen protokol AQMP [15], a to především díky dostupnosti knihoven v různých programovacích jazycích, a také možnost využití open source message brokeru RabbitMQ [16]. Tento protokol zajišťuje sdílení zdrojů mezi programy a aplikacemi pomocí asynchronního předávání zpráv mezi klienty, zajišťuje také implementaci front, zabezpečení a efektivní přenos dat. Nad tímto protokolem byl postaven další protokol, abstrahující komunikaci komponent v implementaci programu v této práci. Tato nadstavba zajišťuje přenos informací v textovém formátu, jako jsou data pro vyhledávání dostupných workerů, data pro aktualizaci jejich databází, obrazová data a odpovědi workerů. Fronty zpráv, přes které komponenty komunikují, se dělí dle jednotlivých workerů a jejich skupin. Jeden hlavní program tedy komunikuje s právě jednou skupinou workerů, kde každý má svou frontu pro příjem dat, a všichni zasílají své výsledky na jednu frontu hlavního programu.



Obrázek 3.3: Sekvenční diagram asynchronní komunikace hlavního programu a workerů, zasílání zpráv a jejich odpovědí.

Jednotlivé zprávy pak vypadají následovně:

Formát zprávy pro hromadné rozesílání (broadcast):
<X><Data>
„<X>“ je byte reprezentující typ zprávy (0 pro vyhledávání workerů, 1 pro přenos databáze osob), „<Data>“ pak reprezentují přenášená data.

Zpráva ke zjištění dostupných workerů, rozeslaná všem, kteří jsou připojeni k dané skupině:
0DISCOVER
Každý worker, připojený k dané skupině, tuto zprávu přijme a odpoví hlavnímu programu. Odpověď, jež je zaslána workerem, obsahuje řetězec s jeho identifikátorem. Formát této odpovědi je **0<Identifikátor>**.

Po zaslání zprávy, která slouží k vyhledávání workerů, hlavní program ještě zašle aktuální databázi osob v následujícím formátu:
1<Seznam>
„<Seznam>“ reprezentuje seznam osob, jehož jednotlivé prvky mají podobu
<Jméno>,<V$_0$>#<V$_1$>#...#<V$_{127}$>\n, kde „<Jméno>“ reprezentuje jméno osoby a „<V$_0$>“ až „<V$_{127}$>“ informace z vektoru rysů.

Obrazová data se pak z hlavního programu k workerům přenáší v následujícím formátu (každá zpráva reprezentuje jeden snímek ve videozáznamu):
<F>,<Data>
„<F>“ je číslo přenášeného snímku ve video sekvenci a „<Data>“ jsou obrazová data v binární formě. Pokud není zasílána sekvence snímků, ale pouze jeden, je číslo snímku buďto 0 v případě běžného zpracovávání jednoho obrázku, nebo -1, v případě, že se zasílá žádost o pouhé získání vektoru rysů. Worker pak na žádost s číslem snímku rovným -1 odpovídá typem zprávy 2, která je popsána níže.

Formát odpovědi hlavnímu programu je pak vypadá následovně:
<Typ>;<Worker>;<Snímek>;<Osoba$_1$>;<Osoba$_2$>;...;<Osoba$_n$>
„<Typ>“ reprezentuje typ odpovědi, který je dán jako 0 pro odpověď na vyhledávání workerů, 1 pro odpověď na zpracování snímků z video sekvence a 2 pro odpověď na zpracování jednoho snímku, pro následné uložení do databáze obličejů.
„<Worker>“ je dané, unikátní jméno workeru, „<Snímek>“ je číslo zpracovaného snímku ve video sekvenci a dále následuje seznam detekovaných a rozpoznaných osob, jehož prvky mají formát **<Pozice>,<Jméno>,<Jistota>,<Rysy>**. „<Pozice>“ je v tomto případě obdélník, jež ohraničuje detekovaný obličej, „<Jméno>“ je jméno rozpoznané osoby, nebo „unknown“ v případě selhání rozpoznávání, „<Jistota>“ je jistota rozpoznání daného obličeje (confidence) a „<Rysy>“ jsou vektor rysů, tedy výstup neuronové sítě. Samotný vektor rysu je zasílán ve formátu **<V$_0$>#<V$_1$>#...#<V$_{127}$>**, kde „<V$_0$>“ až „<V$_{127}$>“ jsou informace z vektoru rysů.

V případě, že nebyl nalezen žádný obličej, worker zasílá:
<Typ>;<Worker>;none;none;0.0;none

Výsledky zpracování videozáznamu se dále ukládají v souborech. Tyto soubory obsahují jednotlivé záznamy ve formátu comma separated values (CSV). Konkrétně jsou využívány dva typy výstupních souborů, jeden pro surový, nezpracovaný výstup workerů a druhý, který obsahuje data zpracovaná do tzv. průchodů. Samotný popis procesu sjednocení výstupních záznamů do průchodů je dále popsán v sekci popisující jeho implementaci.

Záznamy v surovém výstupním souboru mají následující formát:
<Snímek>,<Čas>,<Pozice>,<Jméno>,<Jistota>,<Rysy>\n
Zde jednotlivé části řetězce reprezentují stejná data, jako odpovídající části zmíněné v popisu komunikačního protokolu. Čas pak značí, ve které sekundě videa se snímek vyskytuje.

Záznamy ve zpracovaném výstupním souboru, který obsahuje informace sjednocené do průchodů, vypadají následovně:
<Od_snímku>,<Do_snímku>,<Od_sekundy>,<Do_sekundy>,
<Nejlepší_snímek>,<Jméno>,<Jistota>,<Pozice_na_nejlepším_snímku>\n
V těchto záznamech jednotlivé prvky opět odpovídají stejně pojmenovaným prvkům v popisu komunikačního protokolu. Prvek „<Od_snímku>“ pak reprezentuje, na jakém snímku se osoba v daném průchodu nacházela poprvé, analogicky pak „<Do_snímku>“ je pak číslo snímků, na kterém osoba byla rozpoznána naposledy v daném průchodu.
Časové údaje „<Od_sekundy>“ a „<Do_sekundy>“ značí, které sekundě ve videozáznamu odpovídá první a poslední snímek. „<Nejlepší_snímek>“ reprezentuje číslo snímku, na němž byla hodnota spolehlivosti nejvyšší a „<Pozice_na_nejlepším_snímku>“ pak po-

zici detekovaného obličeje na snímku s nejvyšší spolehlivostí detekce.

Posledním souborem, který program využívá, je databáze osob, která je opět v CSV formátu. Formát jednotlivých záznamů odpovídá seznamu zasílanému komunikačním protokolem, tedy:
<Jméno>,<V_0>#<V_1>#...#<V_{127}>\n
Jednotlivé prvky jsou popsány výše.

3.2 Použité knihovny

Jednotlivé části systému byly implementovány ve dvou programovacích jazycích. Hlavní program s uživatelským rozhraním, který slouží především pro čtení videosouborů a zpracovávání a zobrazení výsledku, je implementován v jazyce Java (verze 8). Program pro zpracování obrazových dat, detekce, rozpoznávání a sledování, je pak implementován v jazyce Python (verze 2.7).

V hlavním programu byly využity knihovny JavaFX pro implementaci grafického rozhraní a OpenCV [24] pro čtení obrazových dat z video a obrazových souborů a také pro práci s obrazovými daty, vektory a maticemi.

Program, jež implementuje workery, využívá následující Python moduly: dlib pro detekci obličejů, zarovnávání a sledování objektů (knihovna dlib byla využita i pro testování alternativní implementace neuronové sítě pro získávání obličejových rysů), OpenCV pro čtení a zpracovávání obrazových dat, modul OpenFace [22], který obsahuje implementaci neuronové sítě pro extrakci rysů, a scikit-learn [36], který poskytuje implementaci klasifikačních algoritmů, jako v této práci využívaný SVM klasifikátor.

K usnadnění nasazení a distribuce byla využita platforma Docker [17], pro kterou byl vytvořen soubor s obrazem systému, který obsahuje kompletní implementaci programu spouštějícího proces detekce, rozpoznávání, sledování a klasifikace osob, tedy workeru. Je tak tedy možné využít i cloudové služby, jako je např. Docker Cloud nebo Amazon AWS. Vzhledem k implementaci programu není nutné na straně cloudových serverů ukládat obrazová data jak ve formě videozáznamu, tak ani ve formě snímků obličejů osob. K akceleraci výpočtů pak lze využít platformu nVidia CUDA, jejíž využití umožňuje implementace neuronové sítě. Tato platforma poskytuje prostředky k distribuci výpočtů na grafickém hardware, je-li kompatibilní s CUDA technologií. Vzhledem k četnému využívání maticových operací během výpočtů při používání i trénování neuronových sítí, tato platforma poskytuje výrazné urychlení.

3.3 Čtení a přenos dat

Než je možné zasílat obrazová data ke zpracování, hlavní proces potřebuje mít k dispozici seznam dostupných workerů. Proces vyhledávání probíhá tak, že hlavní proces zašle zprávu skrze hromadný (broadcast) kanál, kde oznamuje, že vyžaduje odpověď všech workerů v dané skupině. Každý worker, který tuto zprávu přijme, pak odpovídá zprávou obsahující jeho identifikátor. Program si tak udržuje seznam aktivních workerů, který poskytuje potřebné informace k distribuci snímků ke zpracování. Aktualizace tohoto seznamu lze provést v libovolném momentu opětovným zasláním vyhledávací zprávy.

Videozáznamy jsou čteny s využitím knihovny OpenCV, která umožňuje jak získávání jednotlivých snímků, tak navigaci ve videozáznamu, tedy posun na zvolený snímek. Obrazová data jsou uložena v JPEG formátu, a přenesena komunikačním protokolem z hlavního procesu těm procesům, které tato data zpracovávají. Tyto procesy se mohou či nemusí vyskytovat na stejném zařízení jako hlavní program. Díky této distribuci dat je možná relativně vysoká paralelizace, umožňující výrazné urychlení časově náročného zpracovávání snímků, která je omezena pouze rychlostí čtení snímků z videosouboru v hlavním programu. Nejde pouze o paralelizaci více videosouborů, nýbrž o distribuci dat z jednoho záznamu. Jeden záznam je tedy zpracováván paralelně v jednom či více procesech, na jednom či více zařízeních.

Před dalším zpracováním a zasláním se také provádí porovnání s předchozím načteným snímkem. Pokud jsou si snímky podobné natolik, že v obrazu videa nedošlo k významné změně, je snímek přeskočen a načítán další, dokud ke změně nedojde. Hodnota, která určuje vzdálenost dvou matic pixelů snímků jako příliš nízkou k tomu, aby byl snímek relevantní, byla určena na základě experimentů s videem.

Po převodu dat to JPEG formátu jsou snímky asynchronně distribuovány mezi workery, a to tak, že při spuštění zpracovávání se dle počtu připojených workerů stanoví, jak záznam rovnoměrně rozdělit. Poté se pro každý worker vytvoří instance stejnojmenné třídy, která obstarává zasílání přidělených snímků a zpracovávání odpovědí workerů.

O distribuci a předávání výsledků se stará tzv. Dispatcher, jež je v programu reprezentován stejnojmennou třídou. Tento dispatcher zajišťuje distribuci snímků mezi workery a rozdělení videa na jednotlivé sekce, které odpovídají pracovním úkonům workerů. Mimo to se také stará o směrování přijatých odpovědí od workerů k jejich reprezentujícím instancím v hlavním programu, také o udržování stavu zpracovávání aktuální úlohy, tedy stavu zpracovávání vstupního videa, a v případě použití grafického rozhraní také o aktualizaci grafické reprezentace stavu úlohy.

Doručení zpráv ve správném pořadí zajišťují fronty, jejichž implementaci poskytuje implementace protokolu AQMP Java knihovnou RabbitMQ ze strany hlavního programu a Python modulem pika [18] ze strany workerů. Každý worker je identifikován svým unikátním jménem, jak bylo již zmíněno u popisu komunikačního protokolu, a má přiřazenou jednu frontu, identifikovanou odpovídajícím názvem. Zprávy jsou tak zasílány do této fronty, a to po dvojicích. Tedy zatímco se jeden snímek zpracovává, zpráva s daty dalšího snímku je již připravena ve frontě na straně workeru. Po zpracování snímku tak tedy není nutné ještě čekat na přenesení dalšího, pokud rychlost síťového přenosu překoná rychlost zpracovávání, což je dále popsáno a rozebráno v experimentální části této práce.

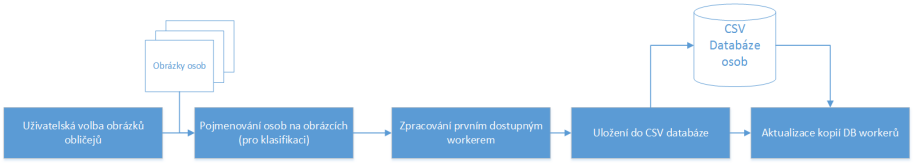
Přijaté odpovědi workerů pak hlavní program zpracovává opět asynchronně. Tyto odpovědi se řadí do jedné společné fronty, ovšem nároky na přenos a jejich zpracování jsou minimální, a fronta tedy slouží pouze jako malá vyrovnávací paměť pro odpovědi, které byly zaslány ve stejný okamžik. Přijaté odpovědi jsou pak distribuovány mezi instance třídy Worker, jejíž identifikátor odpovídá identifikátoru workeru, jež odpověď odeslal. Tato instance třídy se dále stará o přeformátování dat do CSV, aktualizaci statistik zpracovaných snímků a uložení dat do souboru. Každý worker má tedy vlastní dočasný soubor, kde obsah každého souboru reprezentuje právě tu část zpracovaného videozáznamu, která danému workeru byla přidělena. Tyto soubory jsou po dokončení zpracovávání sjednoceny do jednoho souboru, který pak odpovídá celému zpracovanému videozáznamu.

Jelikož se provádí jak asynchronní distribuce snímků, tak i asynchronní zpracovávání odpovědí, jsou použity semafory pro zajištění exkluzivního přístupu ke sdíleným datům.

3.4 Databáze osob

Databáze osob je uchovávána jako jednoduchá tabulka, uložena v souboru ve formátu CSV na straně hlavního programu, a po přenesení také na straně workerů. Každý worker si tak po spojení s hlavním programem tabulku osob aktualizuje a provede přetrénování SVM klasifikátoru, který je dále popsán v sekci popisující klasifikaci a identifikaci osob.

Modifikace a přidávání do této databáze je implementováno tak, že uživatel má možnost přidávat nové osoby dle obrázků, snímků či fotek obsahujících jejich obličeje. Ke každé osobě lze přidat libovolné množství obličejů, které se do databáze uloží ve dvou fázích. Aby nebylo nutné ukládat celá obrazová data a uchovávat tak obrázky osob v systému, stejně jako je přenášet při každé aktualizaci tabulky přes síť, tabulka namísto nich obsahuje vektory rysů, které jsou výrazně méně rozměrné. Toto řešení s sebou nese ovšem jedno omezení a to, že chceme-li přidávat nové osoby do databáze, je třeba mít k dispozici alespoň jeden připojený worker, který se postará o zpracování přiřazených obrázků. Při vložení jména osoby, spolu s obrázky obsahujícími jejich obličeje, se hlavní program připojí k prvnímu dostupnému workeru a postupně zašle všechny obrázky ke zpracování. Poté, co nazpátek přijde odpověď, se teprve do tabulky osob přidá jméno s asociovaným vektorem rysů, získaným ze zaslaného obrázku. Jednou z vlastností těchto vektorů, která je zmiňována i dále při klasifikaci osob a shlukování neznámých osob, je v případě implementací neuronových sítí v této práci, jejich umístění v eukleidovském prostoru. Tedy čím podobnější obličeje dle neuronové sítě jsou, tím nižší je jejich eukleidovská vzdálenost.

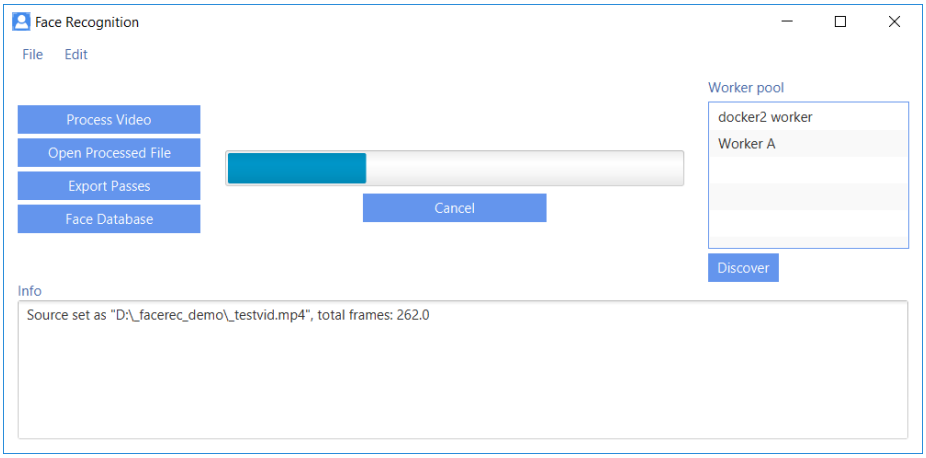


Obrázek 3.4: Diagram postupu vkládání nové osoby do databáze.

Program je možné používat i bez databáze specifikovaných osob a hlavní proces se pak na závěr zpracovávání dat do průchodů bude snažit spojit podobné neznámé osoby dohromady. Tento způsob použití však způsobuje nejen výrazné snížení schopnosti programu rozlišovat jednotlivé osoby, ale také dochází ke snížení přehlednosti vyhledávání osob především mezi více záznamy, jelikož program neznámým osobám automaticky přiřazuje identifikátory, které se mohou a budou u různých souborů lišit. Proces asociace shlukování výskytů neznámých osob je dále popsán v sekci 3.8, která se právě tímto procesem zabývá.

3.5 Uživatelské rozhraní

Implementovaný program poskytuje uživateli dva režimy interakce. Jedním je grafické uživatelské rozhraní a druhým je textové rozhraní. Grafické rozhraní bylo implementováno v prostředí JavaFX 8 s využitím FXML a CSS. Samotný program je tak multiplatformní, omezený pouze nutností přítomnosti JVM a openCV (podporuje Linux, Windows, iOS, Android, ...). Grafické rozhraní (dále jen GUI) se skládá ze tří částí – oken. Hlavní okno aplikace, které je otevřeno při spuštění, lze vidět na následujícím obrázku.

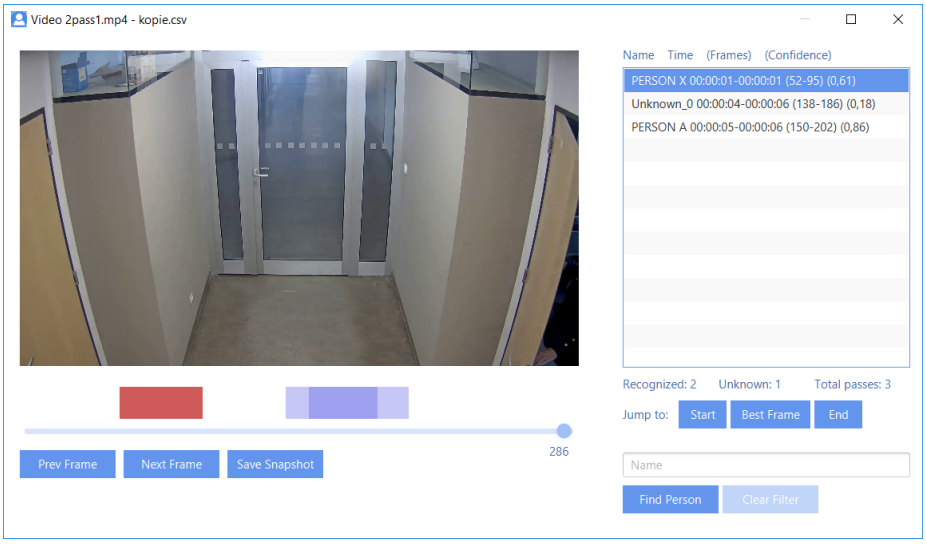


Obrázek 3.5: Hlavní okno uživatelského rozhraní.

Rozhraní poskytuje několik základních prvků pro ovládání programu, jako možnost grafického výběru souboru pro zpracování, otevření již zpracovaného souboru a další. Podstatným prvkem je seznam dostupných workerů a tlačítko pro vyhledání aktivních workerů a aktualizaci seznamu. Graficky je taky znázorněn postup zpracovávání a to s využitím jednoduchého progress baru. Zprávy programu a podstatné informace jsou zobrazovány v odpovídajícím textovém okně – „Info“.

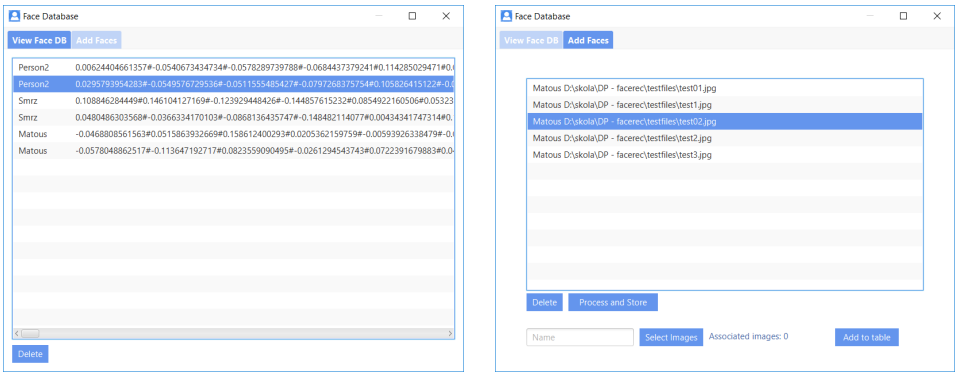
Druhou částí GUI je okno, jež uživateli umožňuje zobrazovat zpracovaná data a k nim přiřazený videosoubor. Přidání videosouboru je nepovinné, avšak umožňuje zobrazení konkrétních snímků. V okně, pod obrazem videa, je mimo nástrojů k navigaci ve videu také grafické znázornění výskytů osob v daném videosouboru, barevně odlišeny jsou také překrývající se výskyty (výraznější barvou) a výskyty vyhledávané osoby či osoby právě zvolené v seznamu (červeně). Seznam výskytů obsahuje jméno osoby a maximální míru jistoty (confidence) v daném průchodu.

Jsou dostupné také nástroje pro vyhledávání osoby v záznamu dle jména a navigace ve zvoleném průchodu osoby (posun ve videu na první, poslední a nejlepší snímek osoby v daném průchodu). Zároveň jsou také zobrazeny souhrnné statistiky: počet rozpoznaných osob, počet průchodů, kde osoby nebyly rozpoznány a počet celkových průchodů.



Obrázek 3.6: Okno pro vyhledávání osob a navigaci ve videozáznamu.

Třetí částí GUI je pak okno, které poskytuje nástroje k zobrazení a editaci databáze osob. Okno je rozděleno právě podle těchto dvou funkcí, a to do dvou záložek. První nabízí seznam osob a jejich vektorů rysů, a také možnost osoby z databáze odstraňovat, zatímco druhá poskytuje komplexnější rozhraní. V druhé záložce může uživatel přidat novou osobu, a to zadáním jména a specifikací jednoho či více obrázků (JPG, PNG, BMP), jež obsahují snímek obličeje osoby. Po volbě jména a obrázků lze osobu přidat do seznamu ke zpracování. Je-li dostupný alespoň jeden worker, lze soubory zaslat ke zpracování. Do databáze se pak po zpracování uloží automaticky.



Obrázek 3.7: Okno pro práci s databází osob a jeho dvě záložky.

Textové uživatelské rozhraní (TUI) umožňuje uživateli řídit program pomocí textových příkazů. TUI poskytuje stejné funkce jako GUI, i když z důvodů asynchronnosti jsou zavedena časová omezení pro vyhledávání dostupných workerů. Detailnější použití TUI je dále popsáno v sekci příloh, tedy až v závěrečné části tohoto dokumentu. Hlavním rozdílem mezi GUI a TUI je však to, že textové rozhraní umožňuje vyhledávání v několika zpracovaných souborech naráz. V TUI je tedy možné zadat několik souborů, a výsledky vyhledávání dané

osoby zahrnují všechny tyto vstupní soubory. Této funkce lze např. využít pro sledování prohybu osoby na více kamerách, či zjišťování místa výskytu osoby v množině záznamů.

3.6 Implementace workerů

Program pro zpracování obrazových dat, tedy worker, je implementován v programovacím jazyce Python 2.7. Využívá knihovny openCV pro zpracování přijatých JPEG snímků a pro škálování obrazu. Tento program může být spuštěn vícekrát na různých zařízeních, a tyto procesy tak mohou kolektivně zpracovávat jeden či více videozáznamů. Paralelizace je, jak již bylo zmíněno v předcházejících sekcích, zajištěna hlavním programem, a tyto procesy pouze přijímají data dostupná v komunikačních frontách, zpracovávají je a odpověď zasílají do hromadné fronty hlavního programu.

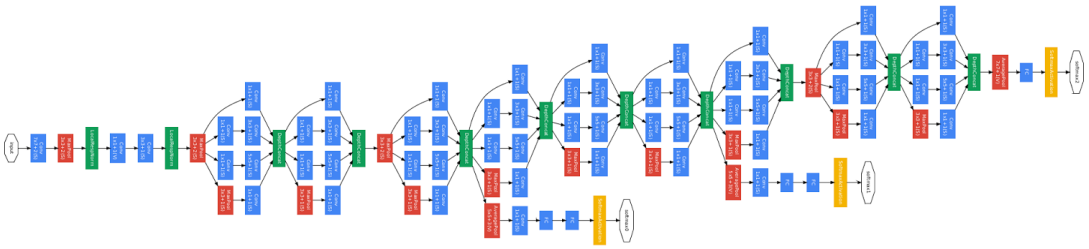
Worker se skládá z pěti hlavních částí.

První je komunikační rozhraní, jež je implementováno s pomocí modulu pika, implementující protokol AQMP pro Python.

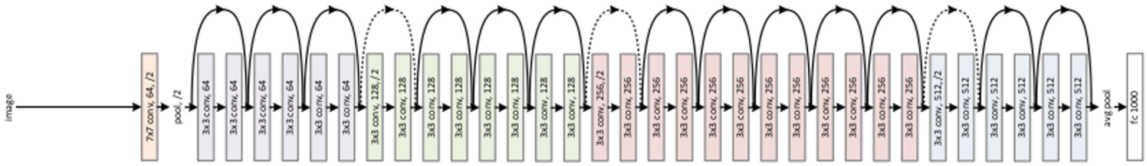
Druhou je detektor obličejů, jehož dvě varianty jsou implementovány buďto v modulu openCV či dlib. Po detekci je ještě provedeno zarovnání obrazu tak, aby výřez detekovaného obličeje odpovídal vstupu konvoluční sítě. Zarovnání je prováděno za pomocí klíčových bodů v obličeji. Konkrétní použitá implementace zarovnávacího algoritmu knihovny dlib využívá 64 klíčových bodů, mezi které patří okolí očí, nosu a celková silueta obličeje.

Třetí částí je algoritmus pro sledování objektů v obrazovém toku, pro který je využito korelačních filtrů implementovaných funkcí correlation_tracker v modulu dlib. Při zpracovávání snímků se prochází všechny detekované obličeje a porovnávají se s obličeji detekovanými ve snímku předchozím. Pokud nějaký z těchto obličejů byl ztracen, použije se předchozí zpracovávaný snímek a incializuje se tracker, který obličej v obraze sleduje. Sledování je ukončeno a tracker je odstraněn v případě, že je detektorem opět nalezen, nebo skóre trackeru klesne pod stanovenou mez. Obličeje se považují za stejné, pokud při porovnání detekovaných obdélníků, jež obličej ohraničují, tyto obdélníky mají poměr obsahu jejich průniku a obsahu jejich sjednocení vyšší, než je stanovená hranice.
Tento poměr se počítá následovně:
Vstup: objekty obdélníků A, B, identifikované levým horním rohem [x0, y0] a pravým dolním rohem [x1, y1].
$S_\cap = max((0, min(A.x1, B.x1) - max(A.x0, B.x0)) * max(0, min(A.y1, B.y1) - max(A.y0, B.y0))$
$S_\cup = A.obsah + B.obsah - S_\cap$
poměr $= S_\cap / S_\cup$

Čtvrtou částí, jež slouží k vytvoření vektoru rysů, je konvoluční neuronová síť, implementovaná dle sítě FaceNet v modulu OpenFace [22], alternativně pak opět konvoluční síť, avšak implementovaná dle sítě ResNet-34 v modulu dlib, která má v implementaci v tomto modulu pouze drobné odlišnosti od sítě původní, avšak zachovává hlavní principy.



Obrázek 3.8: Architektura konvoluční neuronové sítě Facenet-NN2. Modré prvky označují konvoluční vrstvy, červené označují vrstvy typu MaxPool, zelené reprezentují normalizace a slučující operace a oranžové značí softmax prvky. (Převzato z [19]).



Obrázek 3.9: Architektura konvoluční sítě ResNet-34. Jednotlivé konvoluční vrstvy jsou řazeny v sekvenci za sebe, s postupně rostoucím počtem neuronů (64, 128, 256, až 512). (Převzato z [19]).

V programu je využíváno předtrénovaných modelů těchto sítí, které jsou odladěné a testované na větších a filtrovaných datasetech, a dosahují tak lepších výsledků než pouhé trénování na dostupném datasetu LFW, a to i s použitím vlastních dat, jelikož trénování vedlo pouze k velmi mírnému zlepšení na podobných typech videozáznamů, ale zhoršení na jiných, nebo nezlepšilo výsledky vůbec. Bylo tedy možné mírně zlepšit výsledky pro určité záznamy, ale za cenu snížení generalizace sítě. V případě obou použitých neuronových sítí jejich výstup ještě prochází třemi vrstvami v sekvenci, kterými jsou L2 normalizace, embedding a triplet loss. Poslední dvě vrstvy slouží ke spojení výstupu právě do 128-dimenzionálního vektoru a učení s využitím rozdílu mezi stejnými a odlišnými páry obličejů.

Pátou částí je pak klasifikátor, který umožňuje urychlenou klasifikaci obličejů. Je použit pouze k odhadu nejpodobnějšího vektoru rysů z databáze osob, vzhledem k vektoru, jež je výstupem konvoluční sítě. Poté, co je vektor klasifikován, jsou vybrány dvě nejpravděpodobnější třídy, do kterých spadá, tedy dva vektory v databázi osob, jejichž pravděpodobnost je klasifikátorem určena jako první a druhá nejvyšší. Ke klasifikaci je využit SVM klasifikátor dostupný v modulu scikit-learn.

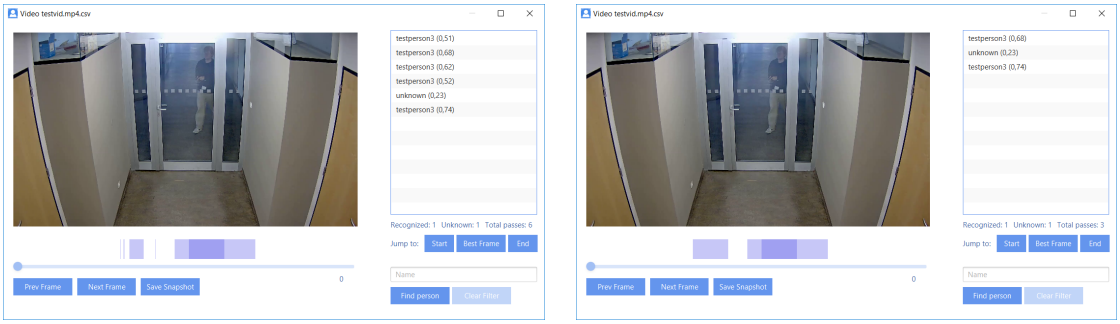
3.7 Sjednocování dat do průchodů

Sjednocování dat ze snímků do jednotlivých průchodů probíhá v několika fázích. Celý proces je možné spustit buďto na již zpracovaném videosouboru, tedy na souboru, jež obsahuje surová data, získaná z videozáznamu, nebo také za běhu programu podle toho, jak data od workerů přicházejí. Sjednocování probíhá především na základě podobnosti obdélníků, které ohraničují detekované obličeje. Využívá se tedy poměru obsahu sjednocení a obsahu průniku obdélníků, podobně jako je využito v části implementující sledovací algoritmy.

Při zahájení se z prvního snímku, kde byl detekován obličej, vytvoří jedna či více instancí objektů, které průchod reprezentují. V programu patří tyto instance třídě Flow. Během čtení výstupních dat workerů, si pak program udržuje v paměti několik částečných, dosud nedokončených průchodů a při přečtení dat nového snímku tyto data porovnává s těmi, jež jsou uloženy v instanci objektu daných průchodů. Pokud není nalezen žádný průchod, do kterého by bylo vhodné detekovanou osobu zařadit, vytvoří se nový. Při každém přečteném snímku pak průchody uložené v dočasné paměti postupně „stárnou“, tedy jednou za snímek se inkrementuje jejich počítadlo neúspěšných zařazení a v případě dosažení prahové hodnoty se tento průchod považuje za ukončený a uloží se.

Při ukládání průchodu se provádí zpracování informací, jež jsou v něm uložené jako výskyty detekovaných osob. Jelikož se může stát, že rozpoznávání v určité části průchodu selže, každý snímek, který je zahrnutý v průchodu, nemusí mít stejné, detekované osoby. V tomto případě se přihlíží na nejčastěji se vyskytující detekovanou osobu v průchodu.

Po dokončení čtení vstupních dat se provede iterace nad všemi uloženými průchody a sjednotí se blízko se vyskytující průchody stejných osob. Někdy totiž může dojít k rozdělení průchodů např. krátkou sekvencí selhání detekce i sledování, avšak lze odhadnout, že pokud se ve videozáznamu vyskytují dva průchody stejné osoby a jsou od sebe pouze několik málo snímků, lze je považovat za průchod jediný. Toto sjednocování tak „vyhladí“ výsledek, a spojí především průchody obtížně detekovatelných osob.



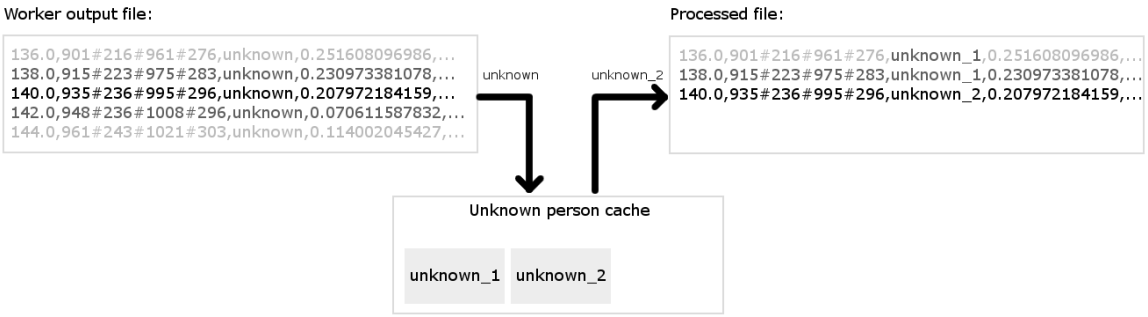
Obrázek 3.10: Změna po aplikaci procesu sjednocování podobných a blízkých průchodů.

3.8 Shlukování neznámých osob

Další funkcí implementovaného programu je shlukování neznámých osob. Toto shlukování se provádí na straně hlavního programu, a to až po zpracování videa. Nezpomaluje tak tedy zpracovávání a je možné jej provádět dostatečně zevrubně. Program při čtení souboru, jež obsahuje surový záznam dat od workerů, udržuje v paměti výskyty neznámých osob

ve formě seznamu a také k nim přiřazené vektory rysů. Konkrétně se u každé osoby ukládá vektor z prvního a posledního snímku, v němž se osoba vyskytuje a také vektor, pro který byla jistota rozpoznání nejnižší, tedy je považován za nejvíce odlišující od osob v databázi.

Při nalezení neznámé osoby v záznamu, program prohledá seznam neznámých osob (při spuštění začíná jako prázdný), a pro každou osobu porovná vektor rysů v aktuálně čteném záznamu, s vektory přiřazenými k osobám v seznamu. Najde-li program takový vektor v seznamu, jehož eukleidovská vzdálenost je menší než stanovený práh, výskyt osoby ve snímku je přiřazen právě k té neznámé osobě v seznamu, jejíž vektor rysů má přiřazen takový vektor. Není-li nalezen žádný vektor splňující tuto podmínku, je vytvořen nový prvek v seznamu - nová neznámá osoba, ke které je pak právě čtený vektor přiřazen.



Obrázek 3.11: Shlukování neznámých osob. Příklad aktuálně čteného záznamu, jež je přiřazen shluku, který je automaticky pojmenován „unknown_2“.

3.9 Rozšiřitelné rozhraní

Díky jednoduchému rozhraní, které je využíváno při komunikaci s workery, je možné tyto workery rozšířit. Program, tedy workery, je možné rozšířit tak, že je dostupný importovatelný Python modul recognizer, který obsahuje funkci getRepFromString(imgstring, skipDetection), která přijímá dva argumenty. Prvním je řetězec obsahující byty obrázku (může být JPG, PNG nebo BMP) a druhým je boolean hodnota, která určuje, zda má být při zpracování snímku vynechána detekce. Návratovou hodnotou je dvojice (vectors, boxes), kde prvním prvkem je pole 128 rozměrných vektorů a druhým je pole ohraničujících obdélníků, které určují, kde se v obraze vyskytuje obličej. Indexy těchto dvou polí musí korespondovat. Tedy obličej i má vektor vectors[i] a vyskytuje se v obdélníku boxes[i]. Vnitřní logika je pak již nezávislá na zbytku programu. Je možné využít Python metody, nebo schopnost Pythonu navázat třeba i na C/C++. Tato modularita umožňuje pomocí tohoto rozhraní využít jinou detekci, zarovnávání, rozpoznávání a sledování objektů, za poskytnutí paralelní platformy. Může být navíc ještě třeba modifikovat výpočet jistoty, tedy nahradit metodu facedb.FaceDB.distToConf, která převádí vzdálenost na jistotu.

Kapitola 4

Experimenty a vyhodnocení

V této kapitole následují experimenty prováděné s implementovaným programem, shrnutí získaných výsledků, jejich porovnání a vyhodnocení. Experimenty jsou rozděleny do dvou sekcí, kde první se zabývá experimenty s jednotlivými workery, tedy s implementací detekce, rozpoznávání, klasifikace a sledování osob, zatímco druhá se pak zabývá experimenty s celým systémem, tedy paralelním zpracováním, urychlením a omezeními systému. Pro stručnost nejsou uváděny veškeré experimenty se systémem, ale jen ty, které buďto vedly k užitečným závěrům, či měly největší vliv na volbu parametrů systému. Na závěr je pak uvedeno celkové vyhodnocení systému a jeho aplikace.

4.1 Detekce, rozpoznávání a klasifikace

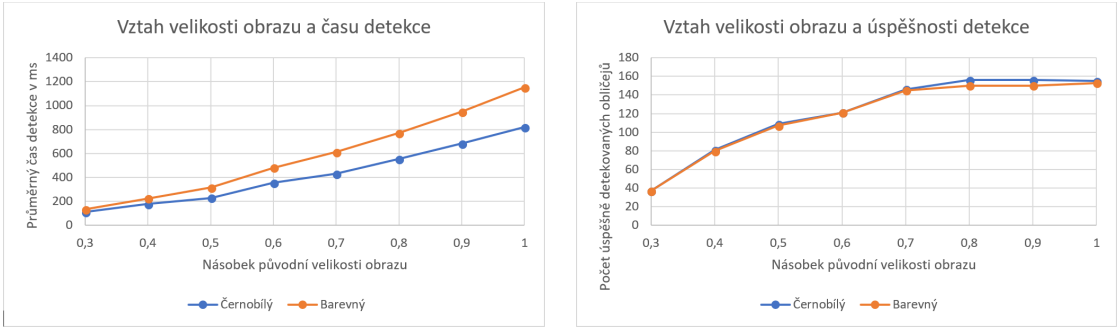
Tato sekce se zabývá experimenty s workery, tedy zpracováváním a informacemi, které přímo souvisí pouze s touto komponentou systému a jsou vyhodnocovány nezávisle na paralelizaci. Experimenty v této sekci, které ověřují výkonnost a rychlost zpracovávání, byly spouštěny na jednom vlákně v jednom procesu. Zařízení, na kterém byly tyto experimenty prováděny, je vybaveno procesorem Intel Core i5-7300HQ s taktem jednoho jádra 2,5 GHz.

Experimenty jsou zde uvedeny v logické návaznosti postupně prováděných změn parametrů systému. Po vyhodnocení poměru rychlosti a výpočetní náročnosti jednotlivých komponent, byla detekční část bezesporu největší zátěží. Nejpomalejší bylo tedy zpracovávání snímků, na nichž byl spouštěn detektor. Právě detekcí a její rychlostí se zabývají první experimenty.

Pro experimentování se systémem byla využita datová sada, která byla vytvořena spojením několika záznamů z bezpečnostních kamer, sledujících jednu z chodeb na fakultě, dále pak malé množství vlastních pořízených záznamů, a také datová sada snímků obličejů LFW.

4.1.1 Experiment 1 - vliv velikosti obrazu na detekci

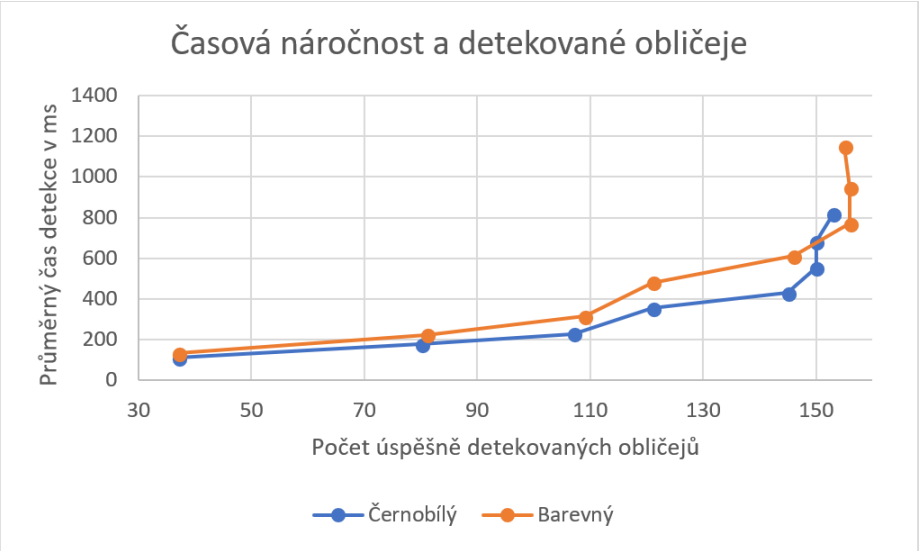
Urychlení detekce je pro konkrétní implementaci detektoru v této práci několik. Na jednotlivých snímcích, nad kterými je detektor spouštěn, je pak možné škálovat vstupní obraz, jehož velikost je po provedení detekce obnovena pro další zpracování a patřičně škálovány také obdélníky vymezující detekované obličeje. Dále je také možné provádět převod barevného obrazu na černobílý. Efekt těchto dvou modifikací lze vidět na následujících grafech. V tomto experimentu odpovídá násobek velikosti obrazu 1,0 videozáznamu, jehož snímky mají rozměry 1920*1080 pixelů. Během experimentování byly testovány i jiné detektory než zvolená implementace (např. frontal face detector z dlib), ale i přes dobré výsledky na LFW datasetu, žádný nedosahoval takových výsledků jako OpenFace implementace na snímcích videa, kde jsou tváře často zabrány z obtížnějších úhlů či za horšího osvětlení.



Obrázek 4.1: Grafy poměru velikosti obrazu k času a úspěšnosti detekce.

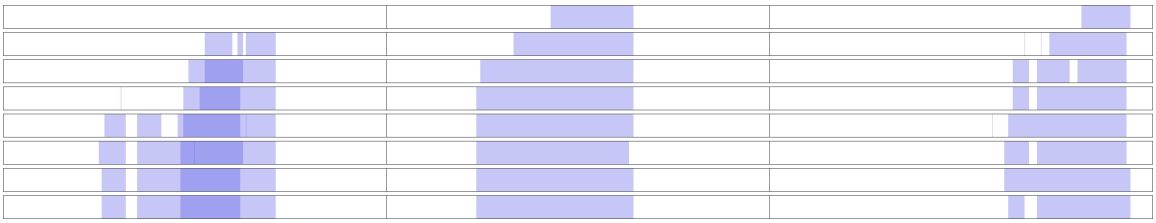
Na grafech z obrázku 4.1 lze vidět křivky zobrazující rostoucí časovou náročnost detekce obličejů s velikostí obrazu, jak pro černobílý obraz, tak pro barevný. Z křivek prvního grafu vyplývá lineární časová složitost detekce v závislosti na velikosti zpracovávaného snímku a lze vidět také rozdíl v časové náročnosti zpracovávání barevného a černobílého obrazu. V druhém grafu pak lze vidět naopak minimální rozdíl mezi černobílým a barevným obrazem ve schopnosti správně detekovat obličej. Detektor, který byl využit, je sice náročný na výpočetní výkon, avšak při zpracovávání obrazu má pouze minimální falešnou pozitivitu. Při experimentování s používanými datovými sadami byla falešná pozitivita detektoru velmi nízká (na videích z chodby na fakultě byl detekován neexistující obličej na 0,09% z celkových snímků). Při zpracovávání videa máme k dispozici několik navazujících snímků a k rozpoznání osoby stačí, aby byl spolehlivě rozpoznán alespoň jeden snímek v této sekvenci. Naopak falešná pozitivita detekce vede k nerozpoznatelným objektům, které ve videu vytváří falešné výskyty neznámých osob.

Následující graf pak zobrazuje spojení předchozích dvou ve formě poměru časové náročnosti detekce a její úspěšnosti. Z následujícího a z předchozích grafů pak vyplývá, že použití černobílého obrazu vede k nižší časové náročnosti za cenu pouze minimálního snížení schopnosti úspěšné detekce.



Obrázek 4.2: Graf zobrazující poměr úspěšnosti detekce a doby zpracování, slučující data z grafů na obrázku 4.1.

Aby bylo možné lépe rozhodnout, jakou maximální velikost obrazu pro detektor zvolit, tedy jak je možné obraz redukovat a stále neztrácet průchody osob, byly konkrétní experimenty analyzovány s využitím grafického rozhraní implementovaného programu. V následujícím obrázku jsou uvedeny jednotlivé experimenty s velikostí obrazu, kde jsou znázorněny jednotlivé snímky, v nichž byl úspěšně detekován obličej.



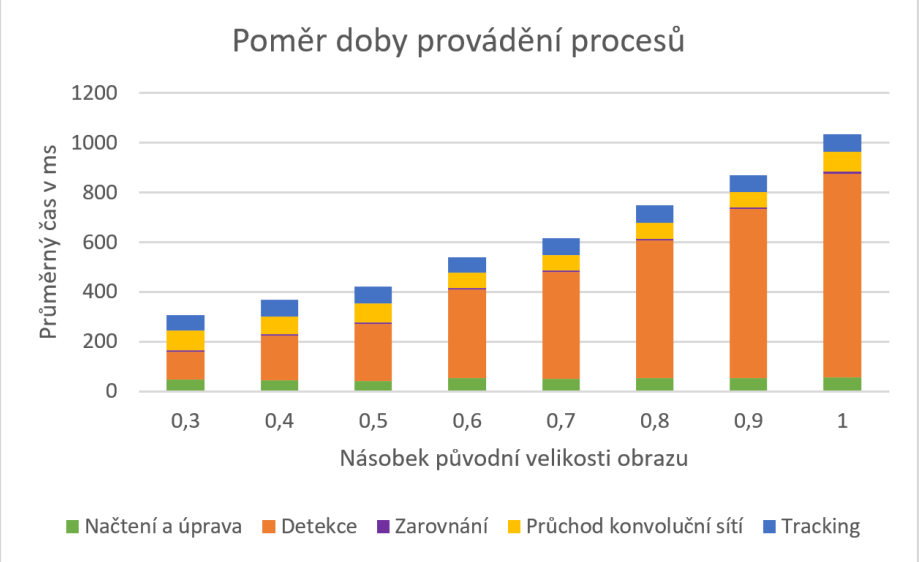
Obrázek 4.3: Detekované snímky v závislosti na velikosti obrazu. Řádek znázorňuje jeden experiment na třech reprezentativních vzorcích, jeden pruh znázorňuje sekvenci snímků a modrá barva pak detekovaný snímek v sekvenci videa. Tmavší modrá barva znázorňuje více vyskytujících se obličejů na jednom snímku. Jednotlivé řádky pak reprezentují experimenty s hodnotami násobku velikosti obrazu 0,3, 0,4, 0,5, 0,6, 0,7, 0,8, 0,9 a 1,0 odshora dolů. Experimenty byly prováděny na větší sadě, avšak pro grafickou reprezentaci byl vybrán pouze demonstrativní vzorek.

Na obrázku lze vidět podstatnou změnu mezi násobky velikosti 0,8 a 0,7, kde dochází k rozdělování souvislého výskytu osoby, a mezi 0,7 a 0,6, kde v dochází ke ztrátě části

informace, avšak stále ne kompletní. Výskyt osoby by byl i v případě násobku 0,6 stále detekován, avšak malý počet snímků by nemusel stačit ke správné identifikaci.

Z těchto experimentů byly určeny hodnoty 0,8 jako dostatečně spolehlivá, avšak náročná možnost, 0,6 jako kompromis mezi rychlostí a náročností a 0,5 jako rychlá, avšak nepřesná varianta. Pro detekci bylo také zvoleno používání černobílého obrazu oproti barevnému.

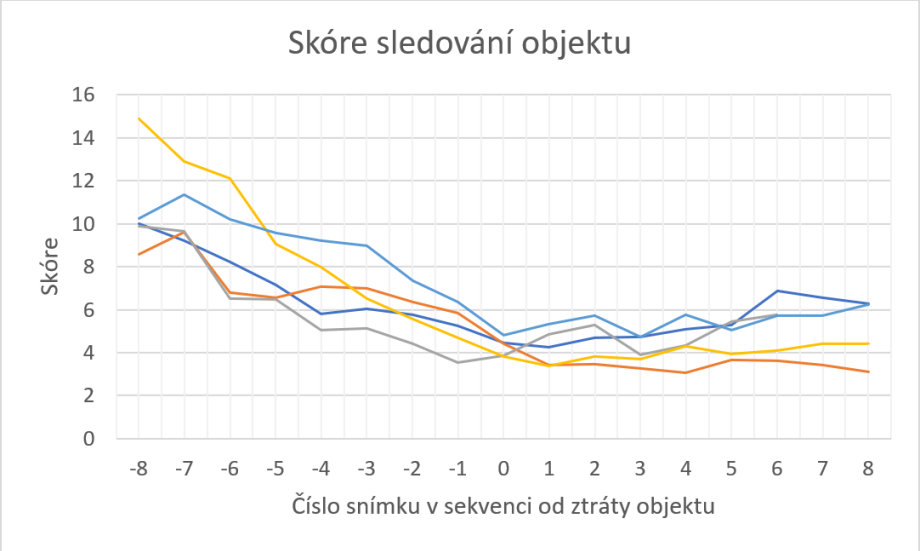
I přes tuto modifikaci však detekce stále zabírá podstatnou část doby zpracovávání snímku, jak lze vidět na následujícím obrázku. Řešením tohoto problému se zabývají následující experimenty.



Obrázek 4.4: Sloupcový graf znázorňující poměr doby zpracovávání jednotlivých částí zpracovávání snímku workerem, kde každý sloupec reprezentuje odlišnou velikost vstupního snímku.

4.1.2 Experiment 2 - sledování objektů

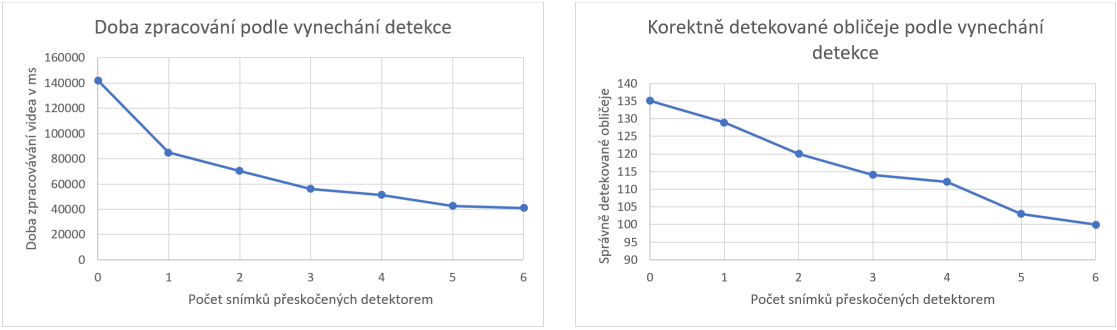
Druhým podstatným experimentem je ověřování vlastností zvolené implementace algoritmu pro sledování objektů v obrazovém toku (dále zmiňován jako tracker). Sledování objektů totiž lze využít nejen při selhání běžné detekce, ale také k urychlení celého procesu zpracovávání videa. Kvůli časové náročnosti je detekce na určitých snímcích vynechávána a pomocí trackeru je pak odhadována pozice objektu, jež byl detekován v předchozích snímcích. Jedním z výstupů trackeru je také skóre, které určuje jistotu toho, zda je stále sledován odpovídající objekt, nebo zda došlo ke ztrátě objektu. Čím vyšší je tato hodnota, tím jistější je, že objekt je stále sledován správně. Chceme-li však určit, zda objekt byl či nebyl ztracen, je třeba stanovit práh. Tento práh je určen na základě experimentů s objekty ve videu. Bylo vybráno několik videí, v nichž se vyskytovaly různé obličeje a ke každému byl po detekci přiřazen tracker. Při ztrátě objektu se tracker ve všech případech choval velmi podobně, tedy tak, jak je zobrazeno v grafu na obrázku 4.5 s využitím pěti reprezentativních vzorků s co nejodlišnějším chováním.



Obrázek 4.5: Graf skóre trackeru pro snímky v okolí snímku, kde došlo ke ztrátě sledovaného objektu. Každá křivka reprezentuje průběh změny skóre jednoho trackeru. Všechny tyto průběhy jsou zarovnány dle snímku, na kterém došlo ke ztrátě sledovaného objektu. Tento snímek je na ose x značen číslem 0.

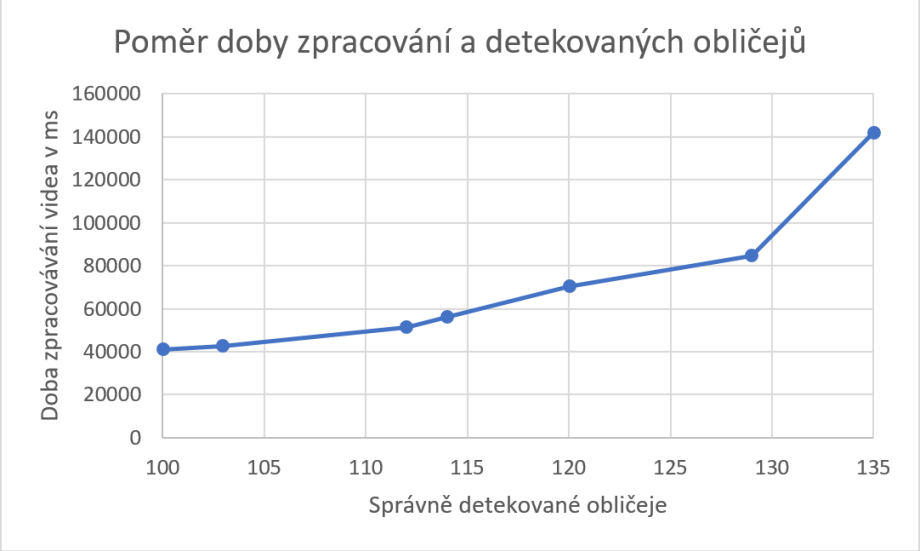
Průměrné skóre trackeru na snímku, na kterém byl objekt ztracen, bylo 4,38, nejvyšší 4,81 a nejnižší 3,82. Na základě naměřených hodnot byl práh úspěšnosti sledování tedy určen jako 4,8.

Po určení prahu již bylo možné experimentovat se spolehlivostí trackeru v kombinaci s detektorem. V tomto experimentu bylo provedeno porovnání doby zpracovávání videozáznamu s počtem snímků, na kterých byla detekce vynechána a nahrazena trackerem. Počet vynechaných snímků pak odpovídá tomu, kolik zpracovávaných snímků bylo ve videosekvenci v cyklu vynecháno, než byl nad snímkem opět spuštěn detektor. Experiment byl prováděn na stejných videozáznamech, jako experiment první, tedy na videu z rozlišením 1920*1080 px a snímkovou frekvencí 15 fps (původně 30, snímková frekvence je však redukována, a pouze každý druhý snímek je zasílán workerům).



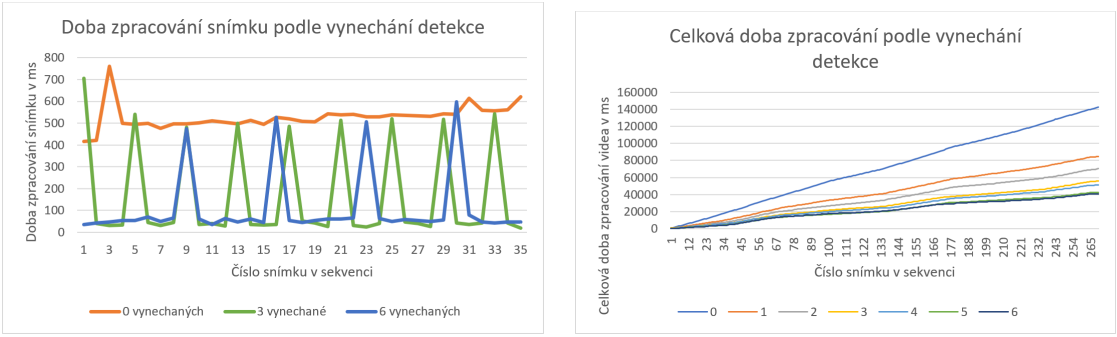
Obrázek 4.6: Grafy zobrazující poměr počtu vynechávaných snímků a doby detekce, resp. počtu správně detekovaných obličejů. Jako referenční hodnota pro druhý graf (vpravo) slouží počet detekovaných obličejů při 0 vynechaných snímcích; tedy nad každým snímkem byl spouštěn detektor.

Z prvního grafu na obrázku 4.6 (vlevo) lze určit, že s nárůstem vynechávaných snímku čas zpracovávání videa exponenciálně klesá, avšak počet korektně detekovaných obličejů klesá pouze lineárně. Sloučení těchto dvou grafů pak lze vidět na obrázku 4.7 ve formě poměru doby zpracování a úspěšně detekovaných obličejů.



Obrázek 4.7: Graf spojující informace z grafů na obrázku 4.6, což je poměr doby zpracování a počtu úspěšně detekovaných obličejů.

Na grafech z obrázků 4.8 pak lze vidět dobu zpracovávání jednotlivých snímků a postupně se zvyšující dobu zpracovávání videa na dvou krátkých videosekvencích.

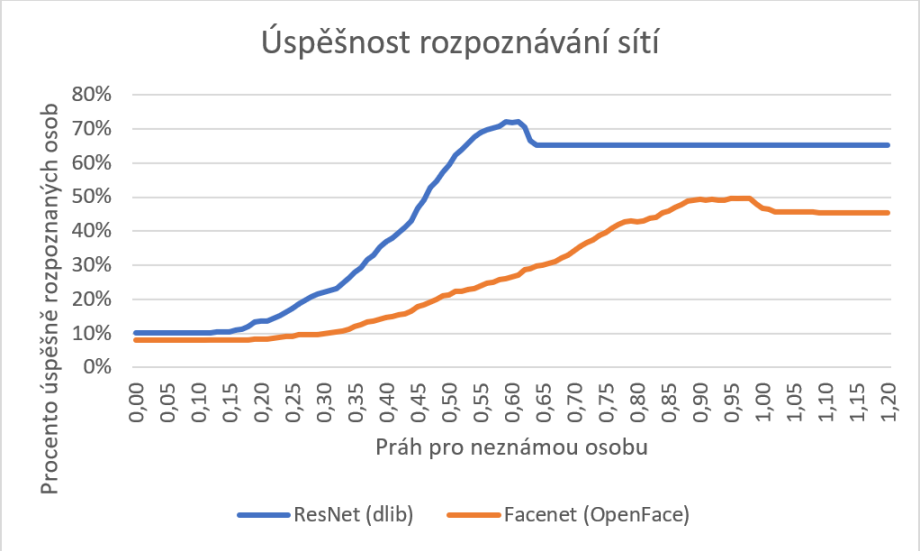


Obrázek 4.8: Grafy zobrazující dobu zpracovávání konkrétních snímku a celé sekvence podle počtu snímků s vynecháním detekce.

Na základě informací získaných v tomto experimentu, bylo využito vynechání detekce pouze ve 3 snímcích v cyklu, jinými slovy: detekce je spouštěna pouze nad každým čtvrtým snímkem.

4.1.3 Experiment 3 - přesnost rozpoznávání a určení neznámé osoby

Dalším experimentem, a zároveň vyhodnocením schopnosti rozpoznávání na vlastní datové sadě, byla schopnost rozpoznávání sítí Facenet a Resnet, tedy jejich implementací v OpenFace, resp. dlib. K testování schopnosti rozpoznávat osoby z videa byla použita vlastní datová sada, obsahující okolo 500 snímků z různých videozáznamů, různých jejich částí a za různých podmínek, jako je osvětlení a úhel pohledu na procházející osoby. K osobám, jež se na těchto snímcích vyskytují, byla v implementovaném programu vytvořena sada rysů obličejů, a to pro každou osobu ze tří různých snímků, kde jeden byl vždy z jiného zdroje než z videozáznamu (webové stránky fakulty, Google images). Na této sadě byla ověřena schopnost rozpoznávání osob pro obě implementace sítí vzhledem k stanovenému prahu vzdálenosti mezi vektory, který určuje, zda je osoba neznámá či není. V případě, že je určen takový nejbližší vektor k detekované osobě, který má vzdálenost od vektoru pro detekovanou osobu větší než stanovaný práh, je osoba považována za neznámou. V měření je zahrnuto také ověřování správnosti určení, zda je osoba neznámá, či ne.

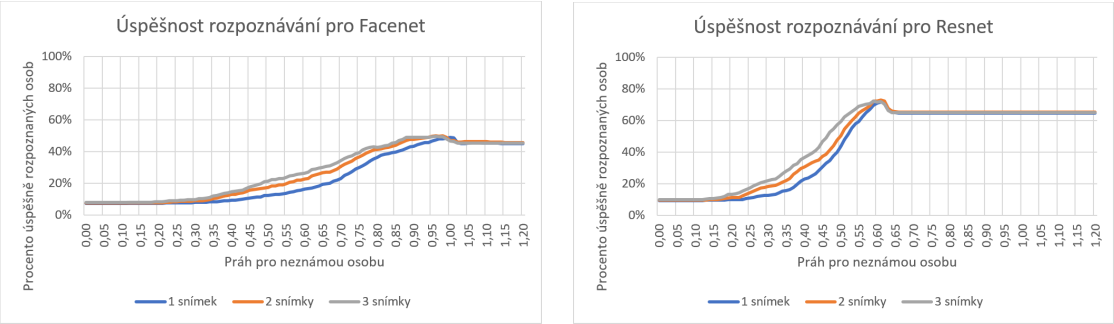


Obrázek 4.9: Graf porovnávající přesnost rozpoznávání sítě Facenet a Resnet s využitím databáze 12 osob. Přesnost po krátkém poklesu zůstává neměnná, protože většina osob v datové sadě byla zároveň v databázi. Relevantními body pro určování prahu jsou však vrcholy křivek, tedy maxima.

Na grafu z obrázku 4.9 lze vidět, že síť ResNet měla vykazovala lepší schopnost rozpoznávání. Mimo to lze také určit vhodný práh pro neznáme osoby, který je okolo 0,59 pro ResNet a 0,9 pro Facenet. Výstup sítě ResNet byl oproti výstupu sítě Facenet však hůře klasifikován při použití SVM klasifikátoru a bylo tedy využito jiného způsobu rozpoznání osoby a to na základě minimální vzdálenosti mezi vektory. Pro tuto metodu naopak Facenet vykazoval nižší přesnost rozpoznávání. Horší přesnost s využitím klasifikátoru byla způsobena především malým množstvím vzorků osob v databázi. Klasifikátor byl pro finální vyhodnocení vyřazen, jelikož se předpokládá, že uživatel nebude mít k dispozici velké množství snímků jedné osoby a také primární činností programu není shlukování neznámých osob. Náročnost hledání minima vzdáleností mezi vektory je velmi nízká: jednotky ms pro 10000 záznamů v databázi osob, okolo 0,01 ms pro desítky záznamů osob v databázi. Byly

ověřovány různé metody při hledání minima, jako minimum ne mezi vektory samotnými, ale mezi rozpoznaným vektorem a průměrnými vektory pro každou osobu. Další ověřovanou metodou bylo také určení shody na základě maximálního počtu vektorů na osobu, jejichž vzdálenosti od rozpoznaného vektoru jsou nižší než stanovená hranice. Tyto modifikace však nevedly k lepším výsledkům, než prosté minimum. Přesnost se sice zlepšila při rozpoznávání u osob, jejichž snímky v databázi obsahují více odlišná data (vektory více rozložené v prostoru), ale snížila se u ostatních.

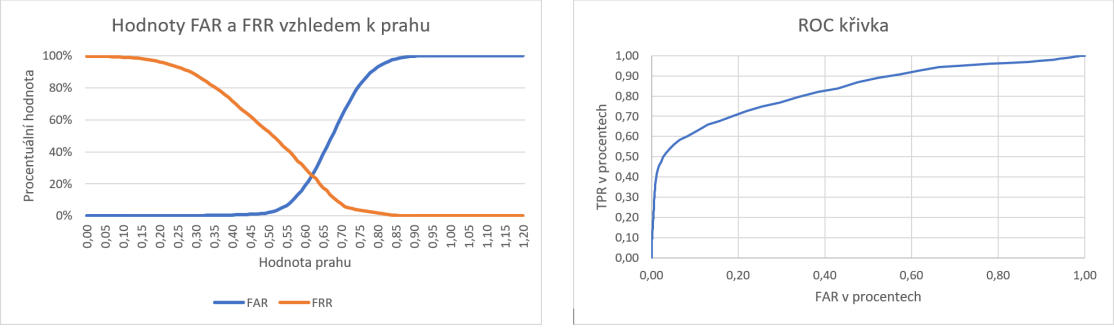
Dále bylo také provedeno porovnání vlivu počtu různých obrázků v databázi osob pro tuto datovou sadu. Byly testovány počty tří, dvou a jednoho snímku pro osobu. Pro každou osobu obsahuje varianta tří obrázků jeden z jiného zdroje než videozáznamu a dva z videozáznamů mimo datovou sadu. Varianta dvou obrázků pak pouze dva z videozáznamů a varianta jednoho obsahuje snímek opět z pouze z videozáznamu. Porovnání bylo provedeno pro obě sítě a lze jej vidět na grafech z obrázku 4.10.



Obrázek 4.10: Grafy porovnávající přesnost rozpoznávání ve vztahu k počtu snímků každé osoby v databázi, s celkovým počtem 12 osob, stejným způsobem jako v grafu na obrázku 4.9.

Úspěšnost rozpoznávání v tomto experimentu dosáhla 72,21%, což může být být vylepšeno při celkovém rozpoznávání vyskytující se osoby, jelikož při spojování výskytů osob do průchodů dojde k vynechání šumu v datech, tedy odlehlé hodnoty jsou zanedbány. Přesnost celého systému (i se spojováním do průchodů) lze vidět v sekci vyhodnocení. Přesnost na datasetu LFW, který obsahuje snímky z lepších úhlů (většinou téměř pravý úhel oproti obličeji), je 92,92% pro použitou implementaci sítě Facenet a 99,13% pro použitou implementaci sítě ResNet.

S prahem bylo dále experimentováno, a byly ověřeny také hodnoty FAR (false acceptance rate), FRR (false rejection rate) a TPR (true positive rate) pro síť ResNet nad datovou sadou z tohoto experimentu. Na rozdíl od předchozích grafů se zde neověřovala úspěšnost rozpoznání s využitím malé databáze, ale byl zkoumán vztah vzdálenosti vektorů rysů a shody osob. Nejsou tedy porovnávány vektory rysů v databázi s vektory rysů ve videu, ale vektory rysů ve videu mezi sebou. Výsledky lze vidět na grafech v obrázku 4.11. Dle protnutí křivek FAR a FRR lze vidět vhodnou hodnotu pro určení prahu.

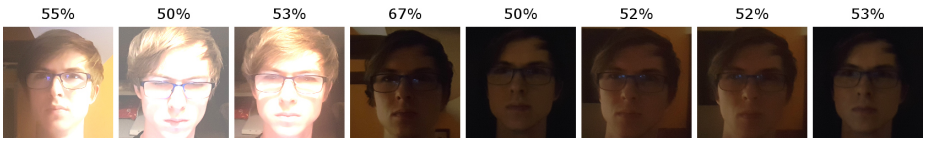


Obrázek 4.11: Grafy hodnot FAR a FRR, jejichž vztah lze vidět na levém grafu a ROC křivka na pravém grafu.

Dále byly také provedeny experimenty se shlukováním neznámých osob, kde osoby byly shlukovány s přesností pohybující se mezi 50 a 60 procenty na této datové sadě i jiných záznamech. Přesnost by bylo možné zvýšit, ovšem na úkor paměti a výkonu. Hlavním účelem tohoto shlukování je však přehlednost výstupního záznamu. Chce-li uživatel vyhledávat neznámou osobu, přidá snímek obličeje této osoby, nebo její vektor rysů do CSV databáze. Poté schopnost rozpoznání dané osoby na této datové sadě roste na 70%.

4.1.4 Experiment 4 - osvětlení a úhly

Prováděny byly také experimenty s osvětlením i úhly kamery vůči osobám, avšak jediným závěrem z těchto experimentů bylo, že v případě sítě ResNet, na které byl experiment prováděn, není limitací schopnost rozpoznávání nýbrž detekce. Pokud byl obličej detekován, pak se ani u špatně osvětlených snímků přesnost sítě nezměnila (pouze zanedbatelné odchylky). Úhel, z nějž byl obličej zabrán, měl na schopnost rozpoznávání negativní vliv, avšak i tento vliv byl eliminován při shlukování blízkých výskytů detekované osoby v reálných videozáznamech. Pro správnou detekci využívaným detektorem je třeba, aby byly viditelné obě oči, ústa a nos.



Obrázek 4.12: Správně detekované a rozpoznané snímky s extrémy v osvětlení a jistota určení správné osoby vyjádřená v procentech.

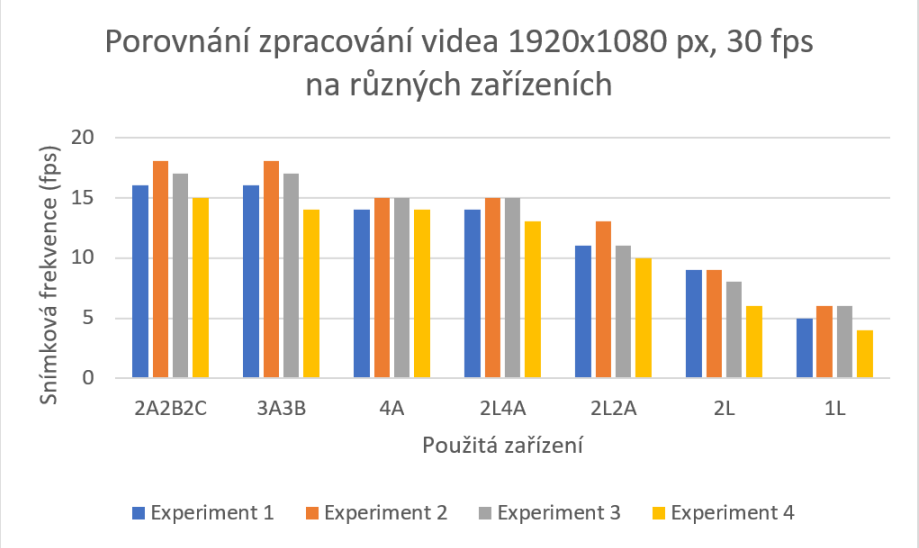
Pro demonstraci jsou na obrázku 4.12 uvedeny hraniční případy, kdy další zesvětlení, resp. ztmavení, vedlo k selhání detekce. Obličeje uvedené na obrázku byly ještě detekovány a správně vyhodnoceny. Experiment probíhal na větší sadě obličejů, avšak z důvodu ochrany osobních dat jsou uvedeny pouze snímky autora. Nad každým snímkem obličeje je také uvedena systémem vyhodnocená jistota rozpoznání. Má-li nejlepší shoda jistotu pod 50%, je osoba podle obličeje označena za neznámou. Tato hodnota také odpovídá eukleidovské vzdálenosti 0,6 mezi vektory rysů; je-li jistota menší než 50%, je vzdálenost větší než 0,6.

4.2 Paralelizace a zpracování

Podstatnou informací je rychlost zpracovávání snímku při paralelizaci, která je zásadní částí implementovaného programu, a to za různých podmínek. K experimentu byla využita čtyři zařízení, jež byly propojeny přes místní síť (1 Gbps ethernet). Zařízení měla následující relevantní parametry:

- L – CPU Intel Core i5-7300HQ @ 2.50 GHz (4 jádra)
- A - CPU Intel Core i5 460M @ 2.53 GHz (4 jádra)
- B - CPU Intel Core i5 3230M @ 2.60 (4 jádra)
- C - CPU Intel Core i3-7100U @ 2.40 GHz (2 jádra)

Zařízení L bylo zároveň tím, na němž byl spuštěn hlavní program. Jednotlivé kombinace jsou v grafu popsány ve formátu XMYNZO, kde X, Y, Z je počet využitých vláken (počet spuštěných workerů) a M, N, O jsou jednotlivá zařízení dle výše uvedeného popisu.



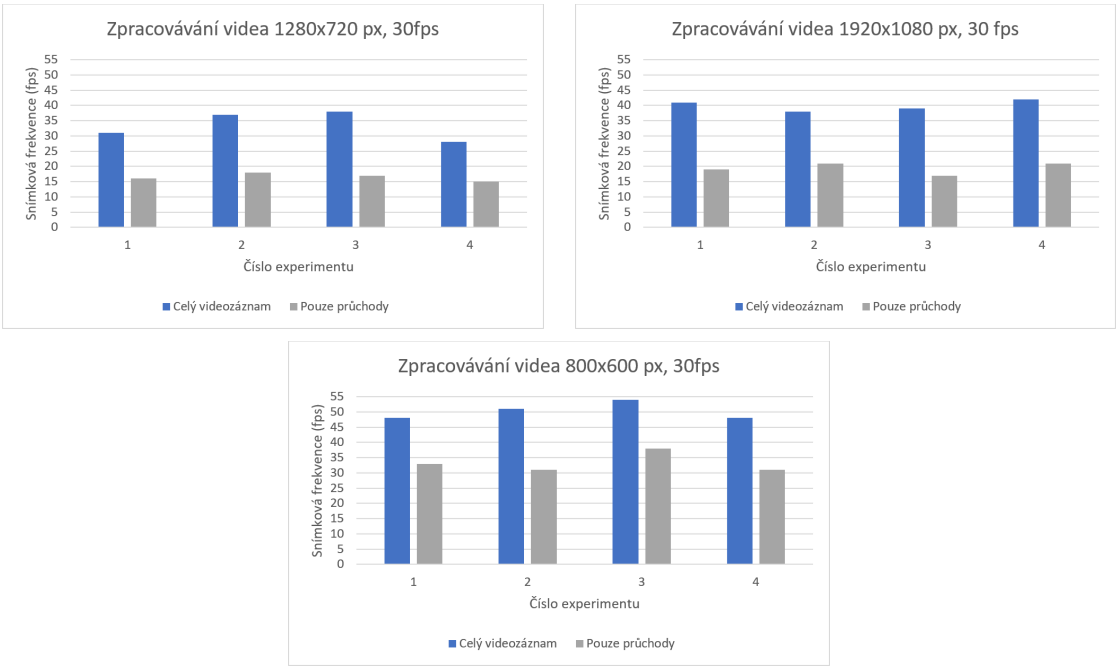
Obrázek 4.13: Graf zobrazující výsledky čtyř experimentů s rychlostí zpracovávání pro každou z různých kombinací zařízení a využitých vláken.

Na grafu z obrázku 4.13 lze vidět čtyři vykonané experimenty pro celkem sedm různých kombinací využitých zařízení. Jednotlivé hodnoty pak značí průměrnou snímkovou frekvenci při zpracovávání videosekvence, avšak pouze pro místa, na kterých se vyskytoval pohyb, tedy docházelo k největší zátěži celého systému; jedná se tedy o minimální snímkovou frekvenci, na kterou rychlost zpracovávání videozáznamu klesla. Při využití tří zařízení pro workery, mimo zařízení L, na kterém byl spuštěn pouze hlavní program, byla minimální snímková frekvence nejvyšší.

Při experimentech bylo zjištěno, že jedním z podstatných omezení je čtení snímků z videosekvence a jejich přenos. I čtení snímků však probíhá paralelně a při využití více než jednoho workeru probíhá rychleji.

Stejné čtyři experimenty byly provedeny pro různou velikost vstupního videa. Pro porovnání je také uvedena průměrná snímková frekvence pro celý videozáznam, který vyl využit

k experimentu. Videozáznamy, tedy experimenty s nimi, které jsou popisovány v grafech na obrázku 4.14, jsou mezi grafy stejné, pouze škálované na uvedenou velikost.



Obrázek 4.14: Grafy zobrazující průměrnou snímkovou frekvenci při provádění čtyř experimentů pro různé velikosti snímků videa. Modře je zobrazena celková průměrná snímková frekvence na záznam, šedě minimální snímková frekvence v úsecích s pohybem osob.

Z experimentů lze vyvodit, že s nárůstem počtu zařízení rychlost zpracovávání roste rychleji, než při nárůstu počtu zpracovávacích vláken na jednotlivých zařízeních. Kombinace využitých vláken a zařízení 2A2B2C byla nejefektivnější. Další přidaná zařízení a vlákna měla již zanedbatelný efekt. Podstatným zjištěním také bylo, že i když kvůli zpomalení způsobeném čtením a přenosem snímků se rychlost mezi 3A3B a 2A2B2C příliš nezmění, lze v tomto případě využít výkonu dodatečného zařízení a dosáhnout tak stále stejné snímkové frekvence při využití efektivnější detekce. Chceme-li pak např. využít lepší detekci a snímky pro detektor neškálovat, přitom však zachovat rychlost zpracovávání, lze pro videozáznam 1920*1080 použít namísto kombinace 3A3B se škálováním 0,6, kombinaci 2A2B2C se škálováním 0,9 bez snížení frekvence zpracovávání.

Během těchto experimentů byla také měřena rychlost přenosu dat a zátěž sítě. Maximální naměřená zátěž sítě je uvedena v tabulce 4.1. Z měření vyplývá, že aby byla jistá sítí neomezená funkcionalita systému, je třeba, aby každý worker byl propojen s hlavním programem tak, aby byla umožněna alespoň rychlost přenosu, která je uvedená v řádku „1 worker“ tabulky 4.1. Jedná se však o maximální naměřené hodnoty, kde maximálních hodnot v tomto řádku dosahovalo zpracovávání pouze jedním workerem, zatímco hodnoty v řádku „Hlavní program“ byly maximální při využití více workerů.

	Uplink	Downlink
1 worker	3,1 MB/s	42,5 kB/s
Hlavní program	5,9 MB/s	121,0 kB/s

Tabulka 4.1: Maximální zátěž sítě pro workery a hlavní program. Uplink značí přenos od hlavního programu k workerům, downlink naopak od workerů ke hlavnímu programu.

4.3 Vyhodnocení na videozáznamech

- Celý systém byl vyhodnocován s následujícími parametry:
- Kombinace zařízení 3A3B (viz podkapitola 4.2).
- Škálování snímku pro detektor: 0,8.
- Použita konvoluční síť ResNet s prahem pro neznámé osoby: 0,57 - byla mírně dána přednost falešné negativitě před falešnou pozitivitou.
- Detekce probíhala na každém 4. snímku zpracovávaném workery.
- Vyhodnocení probíhalo na sadě záznamů z chodeb FIT (každý uváděný soubor obsahuje hodinu videa).

Videosoubor	Správně	Nesprávně	Průchody	Reálné průchody	FPS
20160308140907.mp4	55	26	81	66	31,0
20160308160908.mp4	30	11	41	33	29,7
20160308170908.mp4	18	6	24	20	29,8
20160309094210.mp4	16	3	19	16	30,3
20160310144534.mp4	43	18	61	42	30,9
20180123155213.mp4	53	21	74	68	31,3
20180123195214.mp4	3	1	4	2	32,1
20180123215215.mp4	0	0	0	0	19,2
20180124105222.mp4	55	15	70	56	30,5
20180124145223.mp4	25	13	38	35	29,9
20180124185225.mp4	9	3	12	11	31,0
20180124215226.mp4	2	1	3	2	30,9
Celkem:	309	118	427	351	

Tabulka 4.2: Statistika pro zpracovávaná videa. Tabulka obsahuje název videosouboru, počet správně rozpoznaných osob, špatně rozpoznaných osob, počet detekovaných průchodů, počet skutečných průchodů osob a průměrnou snímkovou frekvenci zpracovávání, tedy rychlost zpracovávání daného videosouboru.

V tabulce 4.2 lze mimo správně a nesprávně rozpoznané osoby vidět také počet jednotlivých detekovaných průchodů osob, který je také jedním z výstupů programu. Skutečné průchody byly kontrolovány manuálně a to i takové průchody, kde se osoba pouze otočila od kamery a poté otočila zpět, jsou považovány za jeden skutečný průchod, zatímco program jej detekuje jako dva. Toto chování by bylo možné změnit použitím detektoru nejen obličeje, ale i celého těla. Takováto detekce však není součástí implementovaného

programu. Celková úspěšnost rozpoznávání na těchto datech je 72,4%, kde pro každý záznam je v databázi zhruba třetina osob, které se v něm vyskytují, aby byly podmínky experimentů co nejpodobnější. Tuto úspěšnost rozpoznávání lze zlepšit, máme-li v databázi co nejvíce těch osob, které se na zpracovávaném videu vyskytují. Např. výsledky na záznamu ze souboru 20160309084210.mp4, pro který byla vytvořena databáze 20 osob, z nichž se 9 v záznamu vyskytuje, lze vidět v matici záměn (tabulka 4.3). Zajímavou odchylkou je záznam 20180123215215.mp4 v tabulce 4.2, pro který je snímková frekvence výrazně nižší než pro ostatní. Rozdíl je způsoben tím, že se jedná o video zaznamenané při nízkém osvětlení, a kamera tak měla nastavenou vysokou citlivost, což způsobovalo tak výrazný šum. Všechny snímky tak byly zpracovávány, jako by na nich byl pohyb, přestože za celou dobu záznamu chodbou neprošla jediná osoba. Tento záznam byl uveden jednak pro demonstraci faktu, že v takovém videu nedochází k falešným detekcím a také znázornění vlivu silného šumu na rychlost systému. Na uvedených záznamech se každá osoba vyskytne průměrně dvakrát, hodnoty tedy odpovídají rozpoznávání variace osob, ne pouze mnoha průchodů jedné osoby.

K falešným detekcím docházelo průměrně 0 až 3 krát na celý hodinový záznam, průměrně dvakrát za záznam. Tyto falešné detekce byly ve zhruba 80% případů rozpoznány jako neznámá osoba a v tabulce jsou všechny započítány jako chybné rozpoznání. Detekci je možné provádět pro různá videa s různě škálovaným obrazem jinak. Např. pro videa, kde kamera zabírá osoby z větší vzdálenosti, je vhodnější obraz méně zmenšovat. V případě, že máme dostupná 4 či více zařízení pro distribuci výpočtu, je vhodné provádět škálování jen minimální (např. pro obraz 1920*1080 px jen poměr 0,9 či 0,8).

Dále bylo zjištěno, že chování systému bylo v uvedených záznamech konzistentní v rámci průchodů stejné osoby. Přesnost určování této osoby tedy byla ve všech záznamech velmi podobná, úspěšnost a chybovost rozpoznání se tedy mezi průchody jedné osoby výrazně nelišila.

Podstatnou informací je také schopnost rozpoznávání osob podle typu osob v databázi. Bylo experimentováno s poměry osob v databázi, které se na záznamech vyskytují a těch, které se na nich nevyskytují. Nejdříve byly v databázi pouze cizí osoby, tedy jednalo se o přesnost rozpoznání pouze neznámých osob, která se pohybovala okolo 80% na záznam, poté poměr 1:1, kde přesnost klesla na cca 72%, a opět stoupla na 76% při použití databáze pouze těch osob, které se na záznamech vyskytují. První pokles je způsoben především volbou prahu, který dává mírnou přednost falešné negativitě před falešnou pozitivitou.

Pokud bychom pak chtěli pouze odlišit počet známých a neznámých osob, přesnost se pohybuje okolo stejných hodnot, jako byly výše uvedeny, je pouze mírně vyšší. Takových výsledků bylo dosaženo, protože častěji dochází k záměně známé osoby za neznámou, než známé osoby za jinou známou osobu. Důvodem takového chování je opět stanovený práh.

Záznamy z tabulky 4.2 byly programem zpracovány, a data, která program potřebuje ke zjišťování informací, byla uložena do datových souborů. Tyto soubory mají pro hodinové záznamy o velikostech 289 až 643 MB velikosti v řádu jednotek MB. Videozáznam ke získání výstupu pak již není potřeba a dochází tak k významné úspoře diskového prostoru. Nad těmito soubory bylo také provedeno vyhledávání jednotlivých osob, pro které má program speciální režim. V tomto režimu prohledává již zpracované soubory a s vyšším prahem hledá pouze jedinou osobu. Vzniklé shluky (výskyty) pak filtruje s přísnějším prahem, podle snímku ve shluku s nejlepší jistotou rozpoznání. Tato metoda byla testována na desíti různých osobách a tyto osoby byly správně rozpoznány v 85,2% případech, což je výrazně vyšší, než přesnost u hromadného zpracovávání. Časová náročnost takového vyhledávání se pohybuje v desítkách až stovkách milisekund na jeden záznam.

		Rozpoznané										
		A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**X**	**N**
Skutečné	**A**	3	0	0	0	0	0	0	0	0	0	0
	B	0	2	0	0	0	0	0	0	0	1	0
	C	1	0	2	0	0	0	0	0	0	0	1
	D	0	0	0	3	0	0	0	0	0	0	1
	E	0	0	0	0	1	0	0	0	0	0	0
	F	0	0	0	0	0	1	0	0	0	0	0
	G	0	0	0	0	0	0	2	0	0	0	0
	H	0	0	0	0	0	0	0	3	0	0	1
	I	0	0	0	0	0	0	0	0	1	0	0
	X	0	0	0	0	0	0	0	0	0	0	0
	N	0	0	0	0	0	0	0	0	0	3	18

Tabulka 4.3: Matice záměn pro 20160309084210.mp4. Jednotlivá písmena reprezentují osoby, „X“ jsou sloučené osoby, jež jsou v databázi, ale ve videu se nevyskytují, „N“ je pak taková osoba, která se ve videu vyskytuje, ale není v databázi.

Při ověřování využívání různých druhů snímku k detekci, jako lze vidět v experimentu 3, bylo zjištěno, že přesnost klesla při využití pouze fotek z webových stránek fakulty. Tento fakt byl znovu ověřován pro datovou sadu uvedenou v této sekci a úspěšnost rozpoznávání při použití pouze jedné fotky z webových stránek pro každou osobu v databázi byla 63,2% při rozpoznávání všech průchodů a 74,0% pro vyhledávání konkrétní osoby se stejnými parametry jako je uvedeno v předchozím odstavci. I fotka ze systému je tedy v testovaných případech dostačující pro nalezení osoby bez zásadní změny spolehlivosti rozpoznávání. Uvedeme-li jako příklad vyhledávání zloděje, pro kterého máme jednu jeho fotku v databázi, systém je schopen tuto osobu nalézt v záznamech bezpečnostních kamer prodejny s přibližně 74% přesností, mají-li kamery podobné parametry, jako v experimentech v této práci, tedy podobnou vzdálenost od procházejících osob a podobné rozlišení obrazu a je-li fotka podobné kvality, jako ty na webových stránkách fakulty.

Jedním ze zásadních vlivů schopnosti rozpoznávání osob je také délka průchodu. Máme-li k dispozici průchod osoby, který se skládá pouze z krátkého zachycení jejího obličeje, přesnost rozpoznávání se s tímto stanoveným prahem pohybuje pouze okolo 60%. Máme-li k dispozici více snímků, přesnost roste až na výše uváděné hodnoty. Během experimentů bylo zjištěno, že nejvyšší přesnosti je dosahováno, máme-li k dispozici alespoň tři snímky v průchodu, na kterých byl detekován obličej, což platí pro valnou většinu průchodů ve videích. Při větším počtu snímků než tři na jeden průchod se přesnost mění již minimálně. Jeden či dva snímky na průchod máme k dispozici pouze v těch okrajových případech, kdy některá z osob se např. podívá ven ze dveří, jež jsou v záznamu umístěny na okraji a rychle se zase otočí.

V případě zjišťování času a doby výskytu konkrétní osoby v záznamu, je poskytována informace o začátku a konci výskytu vzhledem k začátku videa. Jelikož záznamy z různých kamer nemusí poskytovat spolehlivá či žádná data o času pořízení záznamu, stanovení skutečného data a času výskytu osoby lze zajistit dodržováním takového formátu pojmenovávání souborů, který toto umožňuje. Ve výchozím nastavení, i při hromadném zpracovávání videí, program zachovává původní názvy videosouborů, pouze k výstupním souborům přidává příponu.

Pro porovnání s jinými řešeními problému rozpoznávání osob, byly hodnoty úspěšnosti rozpoznávání z podobných záznamů (ze stejného místa a stejných kamer) 81% a 75% pro *Sledování a rozpoznávání lidí na videu* [1] a *Konvoluční neuronové sítě pro bezpečnostní aplikace* [2]. Rychlost zpracovávání byla u první z těchto prací okolo 20 snímků za sekundu. Jelikož nelze přesně replikovat parametry experimentů, jako je databáze osob a v druhém případě i použití konkrétních videozáznamů, výsledky lze porovnávat jen zhruba. Za podobných podmínek experimentů je však patrné, že výsledky těchto dvou prací mají zhruba o 5% až 10% lepší úspěšnost rozpoznávání. Rychlost zpracovávání u tohoto řešení je vyšší než u ostatních, jež byly zmíněny a při vyhledávání pouze jedné konrétní osoby lze pak dosáhnout i podobných výsledků v rámci přesnosti, jako u první z uvedených prací. Ve srovnání s webovým nástrojem OpenFace pro rozpoznávání osob z webkamery, který však slouží jen jako demonstrace, dosahuje v této práci implementovaný systém vyšší přesnosti i rychlosti zpracovávání, a to i s využitím pouze jednoho vlákna pro zpracovávání.

Kapitola 5

Závěr

Na základě teoretického rozboru problematiky byl vytvořen návrh systému pro vyhledávání osob ve videozáznamech. Tento návrh byl implementován a jeho parametry byly určovány na základě experimentů nad datovými sadami statických obrázků (LFW) i videí (záznamů z bezpečnostních kamer). Byl vytvořen systém, který umožňuje paralelizaci zpracovávání, a tedy rozložení zátěže mezi více zařízení, což umožňuje využití i náročnějších procesů jako přesnější detekce obličejů, bez snížení snímkové frekvence zpracovávání za cenu více využitých zařízení. Vytvořený systém je nejen implementací programu pro rozpoznávání osob, ale také platformou pro možné alternativní implementace, poskytující paralelizaci, ukládání a zobrazování dat a informací o průchodech osob.

Systém byl ověřován na datové sadě vytvořené z videí z bezpečnostních kamer na chodbách fakulty, kde při hromadném zpracovávání bylo správně rozpoznáno 72,4% vyskytujících se osob a při ověřování schopnosti vyhledávání jedné konkrétní osoby bylo dosaženo přesnosti 85,2%. Byl-li k dispozici pouze jeden snímek obličeje na osobu, přesnost se pohybovala okolo 63%, resp. 74%.

Průměrná rychlost zpracovávání záznamů se pro video se snímky o velikosti 1920*1080 px pohybuje okolo 30 snímků za sekundu, což je horním limitem systému. Jednotlivé komponenty pak mohou zpracovávat data i pomaleji, bez snížení celkové rychlosti a to s pomocí paralelizace s využitím více zařízení.

Implementovaný systém umožňuje nalézt konkrétní osobu ve videozáznamu, provádět uchovávání veškerých dat potřebných pro vyhledávání osob a zjišťování jejich výskytů ve formě, která je výrazně méně náročná na diskový prostor, než videozáznamy. Systém také dokáže zjišťovat počet výskytů osob ve videozáznamu a určovat neznámé osoby. Redukované množství ukládaných dat umožňuje předzpracování databáze rozměrných záznamů a jejich následnou rychlou analýzu a vyhledávání v nich. Po prvotním zpracování záznamu už veškeré operace nad ním zabírají čas v řádu maximálně stovek milisekund pro hodinové záznamy, na nichž byl systém vyhodnocován. Systém také poskytuje rozšiřitelné rozhraní pro alternativní implementace rozpoznávacích komponent. Databáze osob ani soubory zpracovaných videozáznamů neobsahují samotné snímky, pouze odpovídající vektory rysů a k osobám se tedy neuchovávají obrazová data. Po zpracování záznamů pak lze objemné soubory odložit na méně cenově náročné médium, případně je jinak archivovat.

Systém lze využít např. k vyhledávání opakovaných výskytů osoby, nalezení osob, které by se v monitorovaných prostorech vyskytovat neměly, či rychlé vyhledávání průchodu konkrétní osoby ve velkém množství předzpracovaných záznamů.

Na práci lze dále navázat zdokonalením především rozpoznávacích komponent, a to i na úkor nižší rychlosti zpracovávání, kterou lze kompenzovat právě poskytovanou para-

lelizací. Také by bylo možné optimalizovat čtení a přenos jednotlivých snímků pro snížení omezení celého systému a tak poskytnout i rozsáhlejší možnost distribuovanosti systému, což by vedlo k dalšímu urychlení zpracovávání videozáznamů.

Literatura

[1] ŠAJBOCH, Antonín. *Sledování a rozpoznávání lidí na videu. Brno, 2016.* Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Smrž Pavel.

[2] KIŠŠ, Martin. *Konvoluční neuronové sítě pro bezpečnostní aplikace.* Brno, 2016. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Smrž Pavel.

[3] ILSVRC2017. *ImageNet* [online]. [cit. 28.12.2017]. Dostupné z: http://image-net.org/challenges/LSVRC/2017/results.

[4] Machine Learning is Fun! Part 4: Modern Face Recognition with Deep Learning. *Medium – Read, write and share stories that matter* [online obrázek]. [cit. 10.12.2017]. Dostupné z: https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78.

[5] The Perceptron – Ahmet Taspinar. *Ahmet Taspinar* [online obrázek]. [cit. 14.12.2017]. Dostupné z: http://ataspinar.com/2016/12/22/the-perceptron/.

[6] Matematická biologie učebnice: Koncept umělé neuronové sítě. *Matematická biologie učebnice: Úvod* [online]. [cit. 14.12.2017]. Dostupné z: http://portal.matematickabiologie.cz/index.php?pg=analyza-a-hodnoceni-biologickych-dat--umela-inteligence--neuronove-site-jednotlivy-neuron--uvod-do-neuronovych-siti--koncept-umele-neuronove-site.

[7] How to solve XOR problem with MLP neural network? - *Stack Overflow. Stack Overflow - Where Developers Learn, Share, and Build Careers* [online obrázek]. [cit. 15.12.2017]. Dostupné z: https://stackoverflow.com/questions/6495603/how-to-solve-xor-problem-with-mlp-neural-network/6528548.

[8] Mrázová I. 2014. *Neuronové sítě.* Prezentace k přednášce k předmětu Neuronové sítě (NAIL002), 22-29. Dostupné z: http://ksvi.mff.cuni.cz/~mraz/nn/Neuronove_Site_Prednaska_AM.pdf.

[9] Mitchell, T. M. 1997. Machine Learning. McGraw-Hill. Dostupné z: http://www.cs.cmu.edu/~tom/mlbook.html.

[10] LeCun, Yann; Corinna Cortes; Christopher J.C. Burges. *MNIST handwritten digit database, Yann LeCun, Corinna Cortes and Chris Burges* [online]. [cit. 3.1.2018]. Dostupné z: http://yann.lecun.com/exdb/mnist/.

[11] A Neural Network Playground. *A Neural Network Playground* [online]. [cit. 20.12.2017]. Dostupné z: http://playground.tensorflow.org/.

[12] 2.3. Clustering — scikit-learn 0.19.1 documentation. *scikit-learn: machine learning in Python* [online]. [cit. 7.1.2018]. Dostupné z: http://scikit-learn.org/stable/modules/clustering.html#clustering-performance-evaluation.

[13] Log Analytics With Deep Learning And Machine Learning. *Medium – Read, write and share stories that matter* [online obrázek]. [cit. 10.12.2017]. Dostupné z: https://medium.com/@xenonstack/log-analytics-with-deep-learning-and-machine-learning-20a1891ff70e.

[14] NVIDIA cuDNN | NVIDIA Developer. *NVIDIA Developer* [online]. [cit. 18.12.2017]. Dostupné z: https://developer.nvidia.com/cudnn.

[15] AMQP is the Internet Protocol for Business Messaging [online]. [cit. 01.05.2018]. Dostupné z: http://www.amqp.org/about/what.

[16] RabbitMQ - Messaging that just works [online]. [cit. 03.05.2018]. Dostupné z: https://www.rabbitmq.com/.

[17] Docker - Build, Ship, and Run Any App, Anywhere [online]. [cit. 03.05.2018]. Dostupné z: https://www.docker.com/.

[18] Introduction to Pika — pika 0.10.0 documentation [online]. [cit. 03.05.2018]. Dostupné z: https://pika.readthedocs.io/en/0.10.0/index.html.

[19] Demystifying Parallel and Distributed Deep Learning: An In-Depth Concurrency Analysis. GroundAI - Every ArXiv paper needs to be discussed and rendered beautifully [online]. [cit. 28.12.2017]. Dostupné z: https://www.groundai.com/project/demystifying-parallel-and-distributed-deep-learning-an-in-depth-concurrency-analysis/.

[20] Amin, R.; Gaber, T.; Eltaweel, G.; aj.: *Biometric and Traditional Mobile Authentication Techniques: Overviews and Open Issues*, ročník 70. 02 2014, doi:10.1007/978-3-662-43616-5_16.

[21] Amin, R.; Gaber, T.; Eltaweel, G.; aj.: *Biometric and Traditional Mobile Authentication Techniques: Overviews and Open Issues*, ročník 70. 02 2014, doi:10.1007/978-3-662-43616-5_16.

[22] Amos, B.; Ludwiczuk, B.; Satyanarayanan, M.: OpenFace: A general-purpose face recognition library with mobile applications. Technická zpráva, CMU-CS-16-118, CMU School of Computer Science, 2016.

[23] Benfold, B.; Reid, I.: Stable multi-target tracking in real-time surveillance video. In *CVPR 2011*, June 2011, ISSN 1063-6919, s. 3457–3464, doi:10.1109/CVPR.2011.5995667.

[24] Bradski, G.: The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[25] F.A. Gers, F. C., J. Schmidhuber: Learning to forget: continual prediction with LSTM. *IET Conference Proceedings*, Leden 1999: s. 850–855(5), doi:10.1049/cp:19991218.

[26] Gafurov, D.: A survey of biometric gait recognition: Approaches, security and challenges. 01 2007.

[27] Goodfellow, I.; Bengio, Y.; Courville, A.: *Deep Learning*. MIT Press, 2016, dostupné z: http://www.deeplearningbook.org.

[28] Haxby, J. V.; Ungerleider, L. G.; Horwitz, B.; aj.: Face encoding and recognition in the human brain. *Proceedings of the National Academy of Sciences*, ročník 93, č. 2, 1996: s. 922–927.

[29] Hebb, D.: Organization of Behavior. New York: Science Editions. 1961.

[30] Huang, G. B.; Ramesh, M.; Berg, T.; aj.: Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technická zpráva, Technical Report 07-49, University of Massachusetts, Amherst, 2007.

[31] King, D. E.: Dlib-ml: A Machine Learning Toolkit. *Journal of Machine Learning Research*, ročník 10, 2009: s. 1755–1758.

[32] Mehrotra, K.; Mohan, C. K.; Ranka, S.: *Elements of artificial neural networks*. MIT press, 1997, ISBN 0-262-13328-8.

[33] Nickolls, J.; Buck, I.; Garland, M.; aj.: Scalable Parallel Programming with CUDA. *Queue*, ročník 6, č. 2, Březen 2008: s. 40–53, ISSN 1542-7730.

[34] Parkhi, O. M.; Simonyan, K.; Vedaldi, A.; aj.: A compact and discriminative face track descriptor. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, s. 1693–1700.

[35] Parkhi, O. M.; Vedaldi, A.; Zisserman, A.; aj.: Deep Face Recognition. In *BMVC*, ročník 1, 2015, str. 6.

[36] Pedregosa, F.; Varoquaux, G.; Gramfort, A.; aj.: Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, ročník 12, 2011: s. 2825–2830.

[37] Rosenblatt, F.: Principles of neurodynamics. Perceptrons and the theory of brain mechanisms. Technická zpráva, CORNELL AERONAUTICAL LAB INC BUFFALO NY, 1961.

[38] Russakovsky, O.; Deng, J.; Su, H.; aj.: ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, ročník 115, č. 3, Dec 2015: s. 211–252, ISSN 1573-1405.

[39] Sun, Y.; Chen, Y.; Wang, X.; aj.: Deep Learning Face Representation by Joint Identification-verification. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, Cambridge, MA, USA: MIT Press, 2014, s. 1988–1996, dostupné z: http://dl.acm.org/citation.cfm?id=2969033.2969049.

[40] Taigman, Y.; Yang, M.; Ranzato, M.; aj.: Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, s. 1701–1708.

[41] Tomasi, C.; Kanade, T.: Detection and tracking of point features. 1991.

Přílohy

Seznam příloh

Příloha A

Obsah přiloženého paměťového média

`main-bin/`	spustitelné soubory hlavního programu
`sources/main/`	zdrojové soubory hlavního programu
`sources/worker/`	zdrojové a spustitelné soubory workeru
`install.txt`	návod k instalaci
`poster/`	složka obsahující plakát v PNG a PDF formátu
`worker-image.tar`	Docker image pro worker

Příloha B

Manuál

Před samotným spuštěním programu je třeba provést instalaci. Návod k této instalaci je dostupný v souboru install.txt, který je umístěn na přiloženém paměťovém médiu.
Hlavní program lze spustit následovně:
`java -jar facerec.jar [PARAMETRY]`

Worker lze spustit s využitím platformy Docker:
`docker run --name facerec-container -t -i mjezersky/facerec /bin/bash` (první spuštění)
`docker start facerec-container -i` (Každé další spuštění)

V Docker kontejneru pak:
```
cd  /facerec/
./facerec.py
```

Workery lze konfigurovat změnou souboru „facerec.py“ a hlavní program změnou souboru „facerec.properties“. Je-li následován návod k instalaci, další konfigurace není třeba, vyjma změny identifikátorů workerů, je-li přidáván více než jeden worker.

B.1 Textové rozhraní

K dispozici jsou dva možné režimy použití hlavního programu. Prvním z nich je textové rozhraní. Poté, co je worker spuštěn (a funguje-li připojení k RabbitMQ serveru) lze přidat novou osobu do databáze. Novou osobu přidáme příkazem:
`java -jar facerec.jar -d JMÉNO -i JPG_OBRÁZKY_OSOB`
Poté lze již zpracovat video soubor či video soubory příkazem:
`java -jar facerec.jar -p -i VIDEOSOUBORY`
Výstupní soubory jsou automaticky vytvářeny ve stejném adresáři, jako zdrojové video soubory. Pokud chceme, aby byly výstupní soubory uloženy v jiném adresáři, lze použít parametr `-o CESTA`.
Rozpoznané osoby a jejich průchody lze zobrazit příkazem:
`java -jar facerec.jar -a -i CSV_SOUBORY`
Konkrétní osobu můžeme vyhledávat příkazem:
`java -jar facerec.jar -s JMÉNO -i CSV_SOUBORY`
Pokud bychom chtěli výsledky vyhledávání či zobrazení průchodů uložit ve formátu vhod-

ném pro zpracovávání v jiných programech, lze při použití parametrů -a nebo -s přidat parametr `-e VÝSTUPNÍ_CSV_SOUBOR`, který zajistí, že výsledky budou zároveň vyexportovány do specifikovaného CSV souboru.
V případě, že bychom chtěli použít jiný soubor s databází osob (jak pro ukládání tak pro použití s parametry -p, -a, -s a -l), lze využít parametru `-f DATABÁZOVÝ_CSV_SOUBOR`. Pro vypsání obsahu databáze lze použít příkaz:
`java -jar facerec.jar -l`
Tímto příkazem lze také ověřit, zda byly přidány všechny přidávané obličeje do databáze. Pokud se na obrázku nepodaří rozpoznat obličej, odpovídající záznam do databáze přidán není.

B.2 Grafické rozhraní

V případě, že se hlavní program spustí bez parametrů, spustí se v grafickém režimu. Grafické rozhraní nabízí přehlednější správu a řízení programu. Prvním krokem je nalezení a připojení workerů. To lze provést stisknutím tlačítka „Discover“. V seznamu se pak objeví přopojené workery a jejich identifikátory. Dále lze přidat novou osobu do databáze stisknutím tlačítka „Face Database“, v otevřeném okně databáze pak navigovat do záložky „Add Faces“ a vyplnit pole „Name“ jménem osoby, poté stisknout tlačítko „Select Images“ a vybrat JPG obrázky, které obsahují obličeje dané osoby. Pro přidání osoby do fronty ke zpracování je třeba stisknout tlačítko „Add to Table“. Pokud chceme přidat obrázky z různých zdrojů, lze tuto operaci opakovat vícekrát. Poté už stačí stisknout „Process and Store“, což spustí operaci získání vektorů rysů, které program uloží do CSV databáze. V záložce „View Face DB“ lze zjistit, zda vše proběhlo úspěšně. Obrázky, na kterých nebyl nalezen obličej, do databáze přidány nejsou. V záložce „View Face DB“ také můžeme odstraňovat záznamy z databáze. Okno databáze nyní můžeme zavřít.

Dalším krokem je zpracování videa. Tuto operaci lze provést stisknutím „Process Video“ a volbou video souboru. Výstupní soubory jsou automaticky vytvořeny ve stejném adresáři jako zdrojové video. Pro uložení výstupních souborů do jiného adresáře je třeba použít textové rozhraní.

Detekované a rozpoznané průchody osob lze ze zpracovaných souborů načíst stisknutím „Open Processed File“ a volbou příslušného CSV souboru. Je-li zároveň přítomen také odpovídající video soubor a názvy výstupního CSV souborů či vstupního video souboru nebyly změněny, program přiřadí a otevře video soubor automaticky. Pokud tomu tak není, lze vybrat video soubor ručně, avšak tato operace není povinná. Není-li vybrán video soubor, rozhraní pouze nezobrazí video, avšak přehled průchodů a detekovaných osoby bude stále k dispozici.

V okně s přehledem osob, průchodů a videa lze posouvat zobrazovaný snímek ve videozáznamu (je-li přiřazen), je možné označit průchod osoby a zobrazit první, poslední a nejlepší (s nejvyšší jistotou rozpoznání) snímek v průchodu. Lze také vyhledat konkrétní osobu zadáním jména do pole „Name“ a stistknutím tlačítka „Find Person“.